

difference between file systems and DBMS

(ext 3, ext 4, ntfs, btrfs)

(sqlite, mysql, postgres, mongo, oracle)

Diffⁿ factor

file system

DBMS

defⁿ

File abstraction which helps users to perform manipulation on different types of files.

is software package which allows users to create, update and delete from DB.

data redundancy

in file systems, redundancy is more as compared to DBMS

less redundancy is provided with the help of DBMS as compared to file systems.

data consistency

are less consistent as compared to DBMS.

are more consistent because of less redundancy.

easy to access

are more difficult to access as compared to DBMS.

accessibility is faster as query languages (SQL) have been provided by DBMS.

sharing of data

data is not centrally located as it is scattered at multiple places, sharing info. is difficult.

all the data sets are centrally located and therefore information can be shared easily.

for DB

eg:-

(password protection, security, face, voice detection, fingerprint matching)

no encryption methods were introduced earlier in early days, therefore security was a major concern.

with development, various encryption methods are introduced.

data integrity

There is no mechanism for checking values before entering in the datasets.

Checking constraints are provided so that no wrong data can be entered.

Atomicity

atomicity is absent in file systems.

supports atomicity.

Concurrency

are not supported.

concurrency is provided.

Data isolation

Database

→ next page

data redundancy eg:-

S_rollno name contact no email id DOB branch HOD name HOD contact HOD email

180 - Comps
180 - J.T

$$300 \times 9 = 2700$$

rows col.

Table 1

S_rollno name contact no email id DOB branch id

$$300 \times 6 = 1800$$

1810

Table 2

branch
Branch-id HOD name contact email

$$2 \times 5 = 10$$

data consistency eg:-

If we consider above student, we want to make changes to all 150 rows from ABC to XYZ. If we forget to make changes in two places, we will have 2 values for HOD name, therefore it will result in inconsistent state.

Data isolation → Due to scattered datasets, data can be stored using different file formats and hence data isolation is the problem. Everything is present at one server and therefore datasets are stored in same file format.

Data integrity → Whatever data we enter must be accurate and should follow certain rules. eg:- age of student must be > 20 for admission to an institute.

PAGE NO. _____
DATE ____/____/____

very interesting Atomicity → says all the transactions must be completed successfully or none of them can be completed.

eg:- In banking transaction if there is a real time system error, ~~within~~ the property of atomicity prevents DB from having data in an inconsistent state.

Concurrency → several transactions are trying to access same data item at the same time.

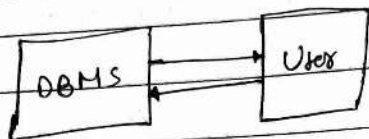
DB
eg:- ~~can~~ can use semaphores as a locking mechanism. The problem is similar to readers-writers problem from O.S.

diff. between semaphores and mutex
→ mutex is a locking mechanism
semaphore is a signalling mechanism.

Architecture of DBMS

It describes how DB components are organized and integrated. Types of DB architecture:-

1 tier arch.

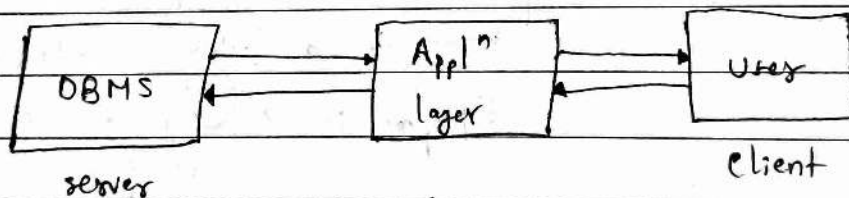


User is directly connected to DBMS and therefore all the changes can be made directly.

Adv. → simple design

Disadv. → Security is a problem, any user can come and make changes.

2 tier arch.

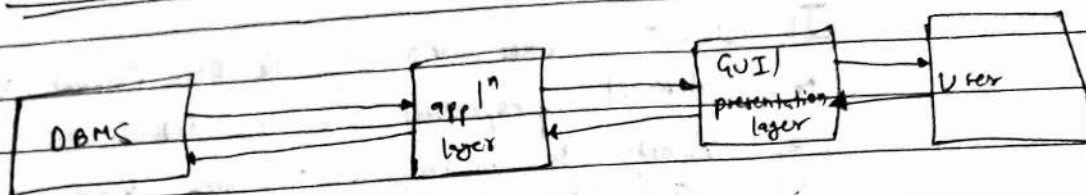


User has to communicate to the applⁿ layer with DBMS.

→ provides set of programs for commⁿ. (eg:-
Connectivity, ODBC - Open Database Connectivity
↳ ASP-net)

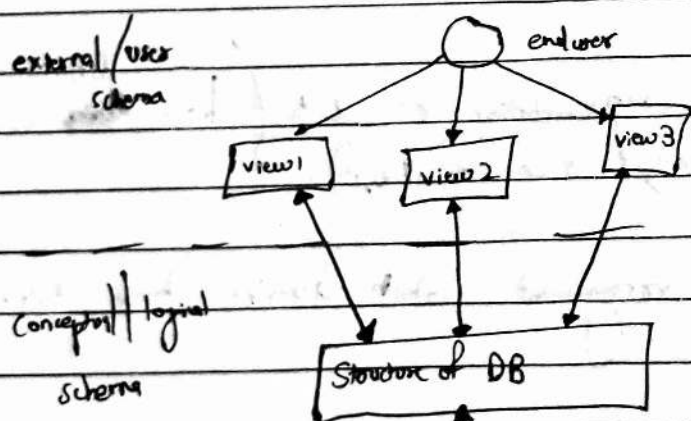
Adv. → security is more as compared to 1 tier arch.

3 tier arch.



User has to communicate with GUI with the DB. User is using various apps as a presentation layer and request is given to the appⁿ layer which will contact DBMS to provide an arch.

Three level architecture or Three views of database

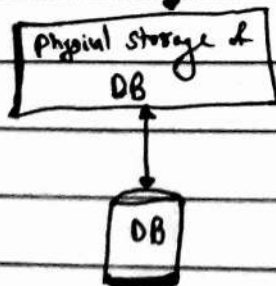


external / user
schema

request is send
by external schema.
response is send
by conceptual
schema.

conceptual /
logical
mapping

Physical / Internal
mapping



DATE / /

external schema / user's view

It describes user's view for the DB. Different users will have different perspectives for the DB.

eg. - According to university, they keep views such as Library and Student (Class advisor) view, with different attributes for both these views.

conceptual / logical schema

It defines structure of DB (no. of entities, attributes, datatypes and data structures).

Physical schema / internal schema

It defines physical representation of DB (how ~~data~~ DB is being stored in a computer, size of DB, etc.).

Schema → An arrangement which describes how data values are stored.

Data independence

It is an ability to modify one level without affecting next higher level.
It gets divided into 2 parts:-

logical
data independence

physical
data independence

logical data independence → it provides an ability to change structure of DB without affecting user's view.

eg:- we can add 'n' no. of records to table, only when info. is related to that particular table.

physical data independence → it provides ability to change structure of DB without affecting logical or user's view of DB.

eg:- moving a file from one location to another (physical storage location) without changing the structure of DB.

Users involved in DB

DBA (Database Administrator): A person (group of persons) centrally located having wide view required to run DB smoothly.

Such a person is granted as DB administrator.

Role performed by DBA:

- 1) Develops conceptual schema \rightarrow defines structure of DB
- 2) Decides which DB to use
 - \rightarrow to create custom inhouse or use ready made tool.
- 3) Defining DB and loading content
 - \rightarrow Actual implementation place.

4) Assists and supervising a 99^{th} access

- \rightarrow helps programmer to write programs

5) Deciding Data structure

- \rightarrow Continuous improvement for changing DB.

6) Backup and recovery

- \rightarrow recover from failure with backups.

7) Integrity

- \rightarrow check data correctness in the DB.

2) DB Designer

3) Security Officer

4) End user

DATE / /

DB designers → helps DBA in designing conceptual schemas.
their job is to meet users and find out their requirements.
they will check whether all the user requirements are fulfilled or not.

Security officers → prevents unauthorized access to the DB.

End user → occasionally accessing database
eg: Principal of a college

Naive/parameteric users → frequently accessing DB
eg: - dealer working at ATM transaction

Sophisticated users → learning from previously developed DBMS tools
eg: and modifying them.
eg: - business analysts, engineers

Standalone users → use ready made DBMS tools.
eg: - librarian of a college

Characteristics of DB

① self describing nature of DB and metadata

metadata → data about data or data dictionary.

keyword :- user manual for the DB.

eg: - data is acquired from multiple sources hence self describing nature and metadata is required.

② Data independence.

encryption

different verification methods

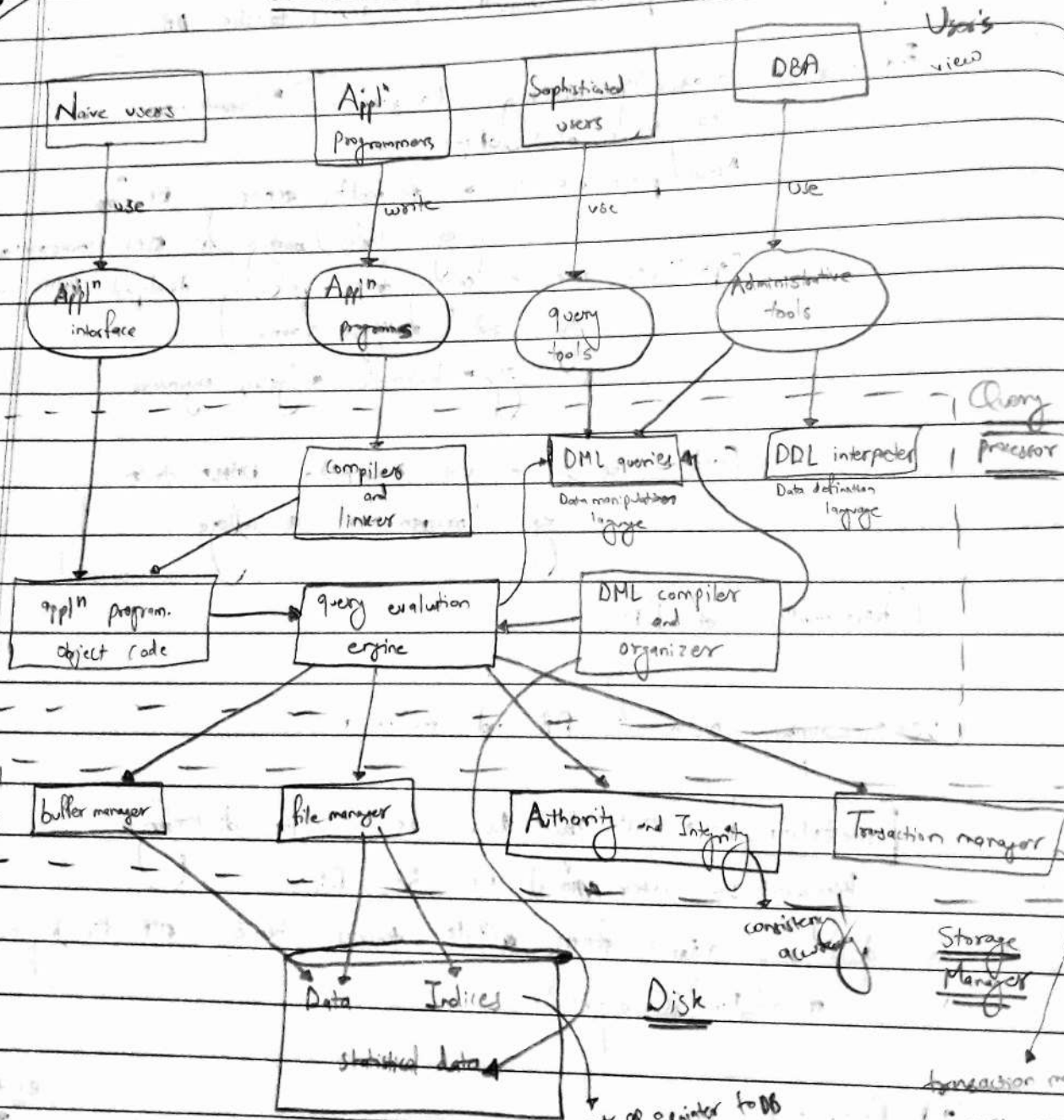
③ Multiple views of DB and security

keyword :- diff. users going to use DB differently.

④ Data Persistence
Data stored in DB remains as it is even if power failure occurs or system gets crashed.

DBMS System Architecture

100%



like crashing faults in the system

Entity Relationship Data Model

• ER Enterprise → nothing but an organization. eg- hospital, bank, college, railway reservation system.

• Entity → any real world object is termed as entity.
(person, place, event, thing).

• Attributes → Special characteristics which describe entity in detail.
eg- student roll-no, name, age, address.

• Entity type → collection of entities having common attribute.
eg- student information.

• Entity set → collection of entities (one or more than one) having same entity type, termed as entity set.

(Subset of entities)

• Value → any information or data stored in a table is termed as value.

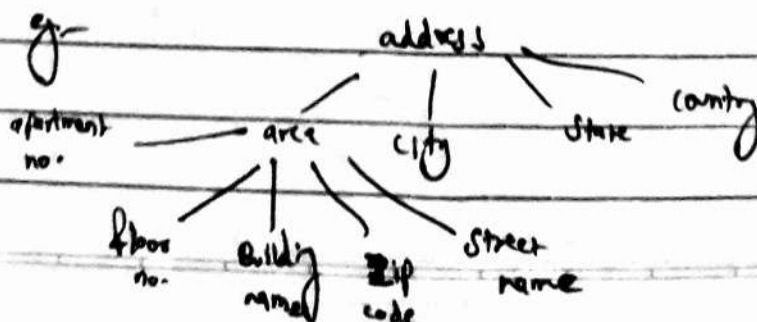
• Domain → set of values for particular attribute termed as domain or value set.

Types of attributes

• Simple vs Composite

↳ this attribute cannot be divided further. eg:- color of car, age, roll no. of student.

Composite → attribute which can further be divided in subparts.



- Single vs. multivalued
 ↳ the attribute which will have only a single value.
 eg:- branch of student, roll no., age, height.

multivalued → attribute which can have multiple values.

eg:- name,
 (first name, middle name, surname)
 email id,
 contact no.

- stored vs. derived

↳ ~~with~~ 'age' attribute can be calculated if we know
 D.O.B of the student and current ~~student~~ date.
 And therefore age is termed as derived attribute,
 which gets derived from one stored attribute
 i.e. D.O.B.

- Null value attribute

Two cases:-

- ① Value does not exist
- ② Value exists, but ~~is~~ not known.

Primary key cannot be empty ^{or} null

PAGE NO.	
DATE	/ /

Keys in DBMS

Key is an attribute (or set of attributes) created in such a way that all records of a table are identified uniquely.

- **super key** → super set of all the keys which will give us uniqueness.
- **Candidate key** → minimal super key (minimal → minimum no. are used to make the key, by removing redundancy).
- **primary key** → any candidate key can be chosen by the developer for unique identification acts as a primary key.
- **secondary/alternate key** → excluding primary key, all other candidate keys are alternate keys.
- **foreign key** → primary key of one table acts as a normal colⁿ in second table.

employee data

	empid	fname	lname	sal	DOB	DOI	passport no	Dept-Id
foreign key → dept-id	1	Priyanka	bbb	75k	13/2/90	1/2/12	-123-	10
	2	Ankita	ccc	80k	2/12/91	1/2/12	-456-	20
primary key → emp-id	3	Priyanka	bbb	85k	13/2/90	4/2/13	-789-	30
	4	Tejanvi	aaa	80k	15/6/85	5/2/15	-023-	10

	Dept-Id	Dept name
Dept data	10	Sales
	20	Marketing
	30	accounts

✓ → accepted Candidate keys
 ⊗ → removed these from Candidate keys

fname, sal
lname, sal

Super keys: - ① emp-id ② emp-id, fname ③ emp-id, lname ④ fname, lname, sal
 ⑤ passport no. ⑥ passport no, dept-Id

Q.	Book-id	name	author
	B ₁	XYZ	A ₁
	B ₂	ABC	A ₁
	B ₃	XYZ	A ₂
	B ₄	ABC	A ₃
	B ₅	LSP	A ₁
	B ₆	ABC	A ₃

Super keys \rightarrow book-id,
 book-id, name,
 book-id, author

Prime, author
 book-id, name, author

Candidate keys \rightarrow book-id,
 name, author

Primary key \rightarrow book-id secondary key \rightarrow name, author

Types of entity sets

① Strong entity set

② Weak entity set

① Strong entity set \rightarrow if an entity set has a primary key attribute, it is termed as strong entity set.

eg:- student (whose roll no. can be primary key)

② Weak entity set \rightarrow if an entity set does not have any primary key, it is termed as weak entity set.

eg:- table (shape, size, color, height) (No primary key)

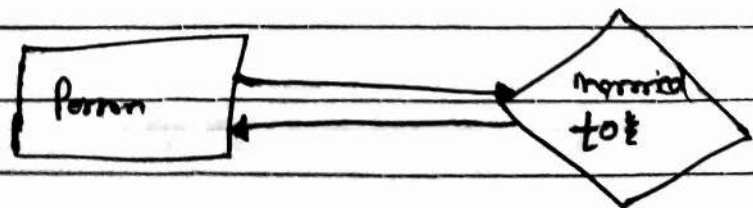
Relationships

It describes an association among several entities.

Degree of relationship

The no. of entities involved in commⁿ describes the degree of relationships.

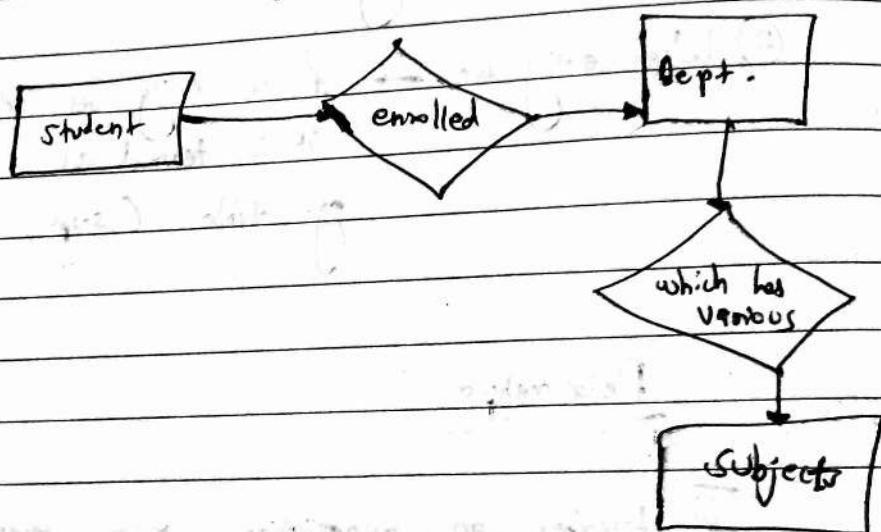
Unary relationship set \rightarrow In this only one entity set participates in the commⁿ.



Binary relationship set \rightarrow In which two entity sets are involved.



Ternary relationship set \rightarrow Student enrolled in department, which has various subjects.



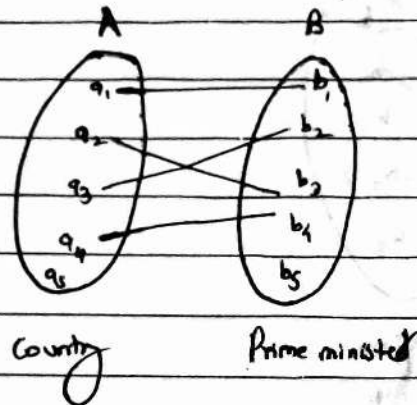
'Many to many' set \rightarrow More than 3 entity sets are involved.
 e.g. prof. teaches in school, college and class.

Can we have zero comm?
 Yes

Mapping constraints / cardinality

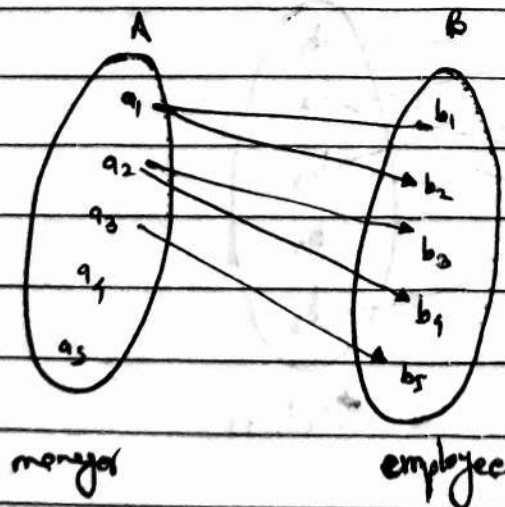
① one to one (1:1)

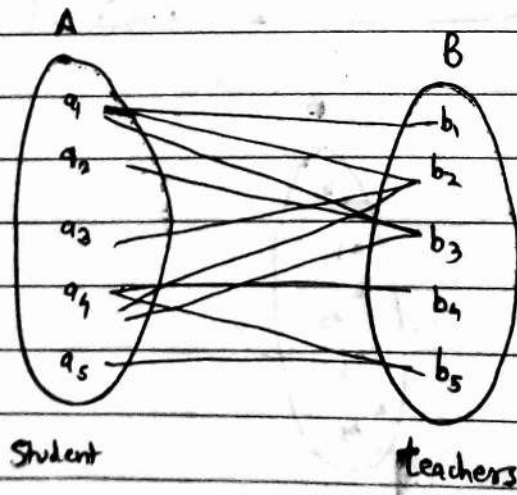
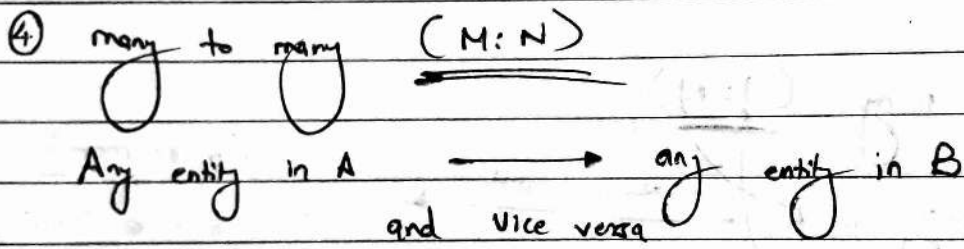
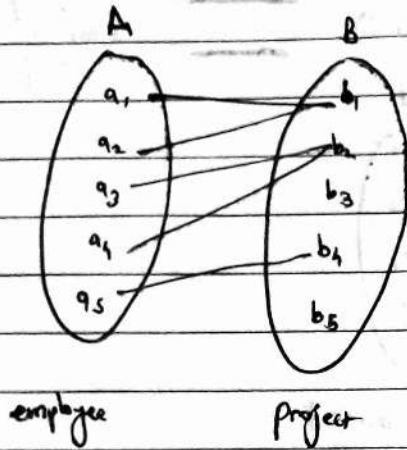
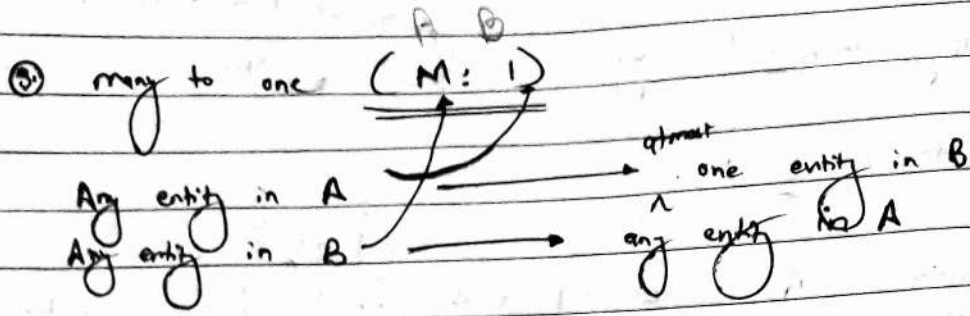
Any entity in A is associated with atmost 1 entity in B.
 Any entity in B is associated with atmost 1 entity in A.



② one to many (1:N)

Any entity in A is associated with any no. of entities in B.
 Any entity in B is associated with atmost 1 entity in A.

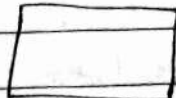


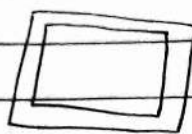


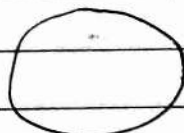
ER model


It is entity-relⁿ diagram which describes logical representation of DB.

Symbols used in ER diagram are as follows:-

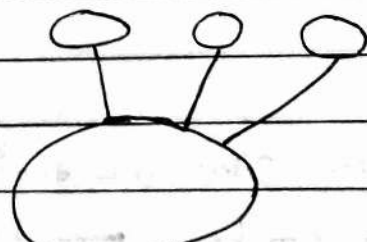
①  → strong entity set

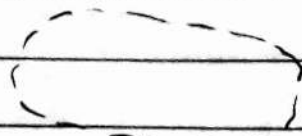
②  → weak entity set

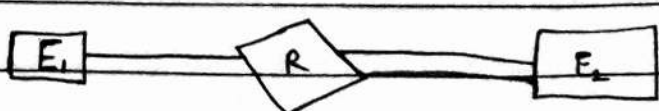
③  → ~~Primary key~~ attribute

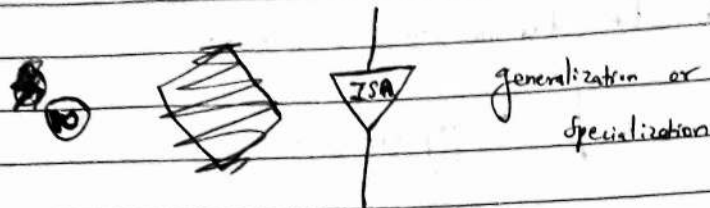
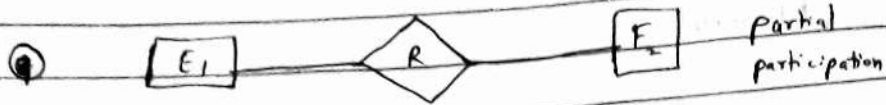
④  → Primary key attribute

⑤  → multivalued

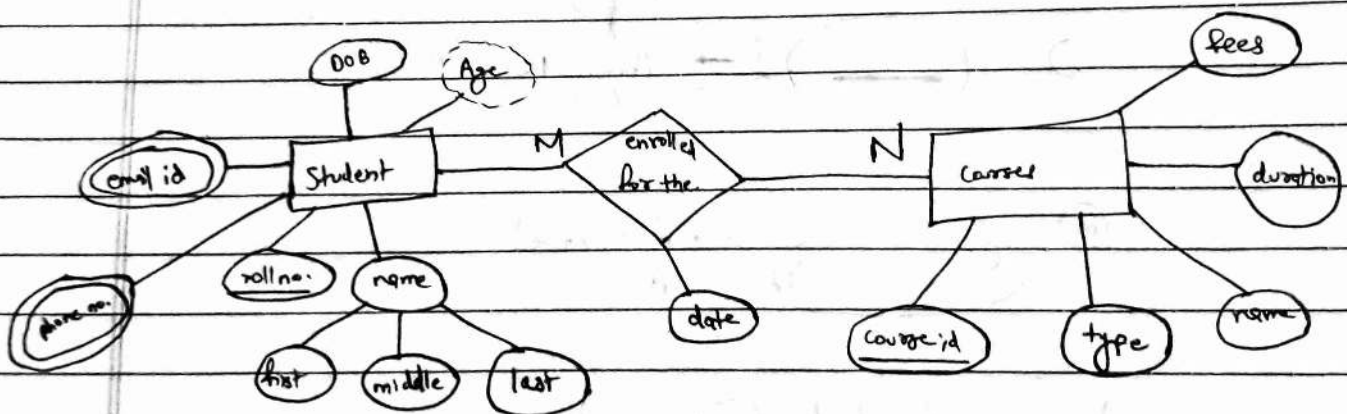
⑥  → Composite attr.

⑦  → derived attr.

⑧  Total participation



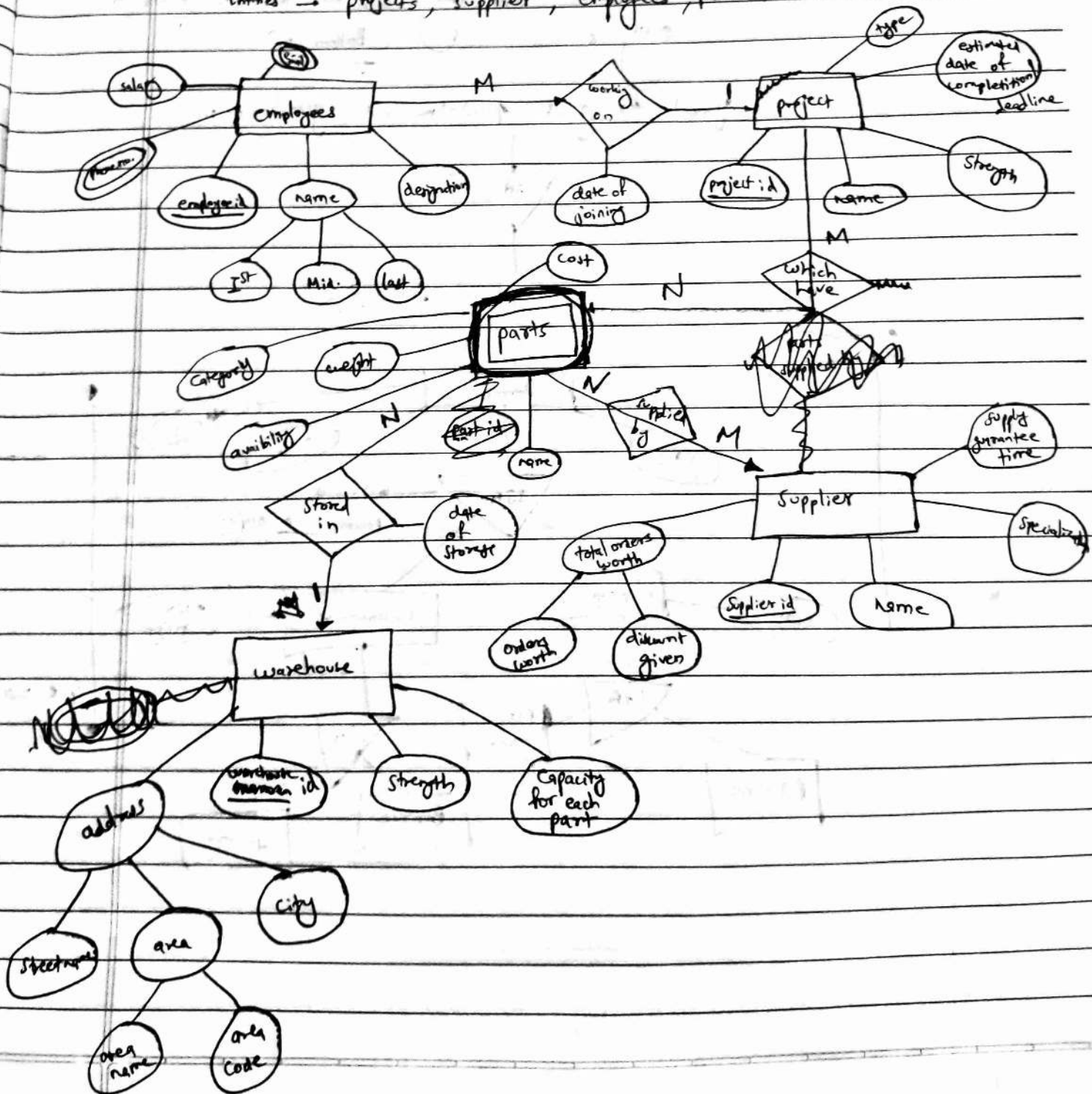
Q. Draw ER diagram showing students enrolled for the courses



M:N as many one student can enroll for many courses
one course can have many students enrolled for the course.

Q. Draw ER diagram of a manufacturing company which records info. about projects. Projects have many parts which are supplied by the supplier. Company maintains warehouse in which diff. parts are stored. Various employees are working on the project.

Entities → projects, supplier, employees, parts, warehouse



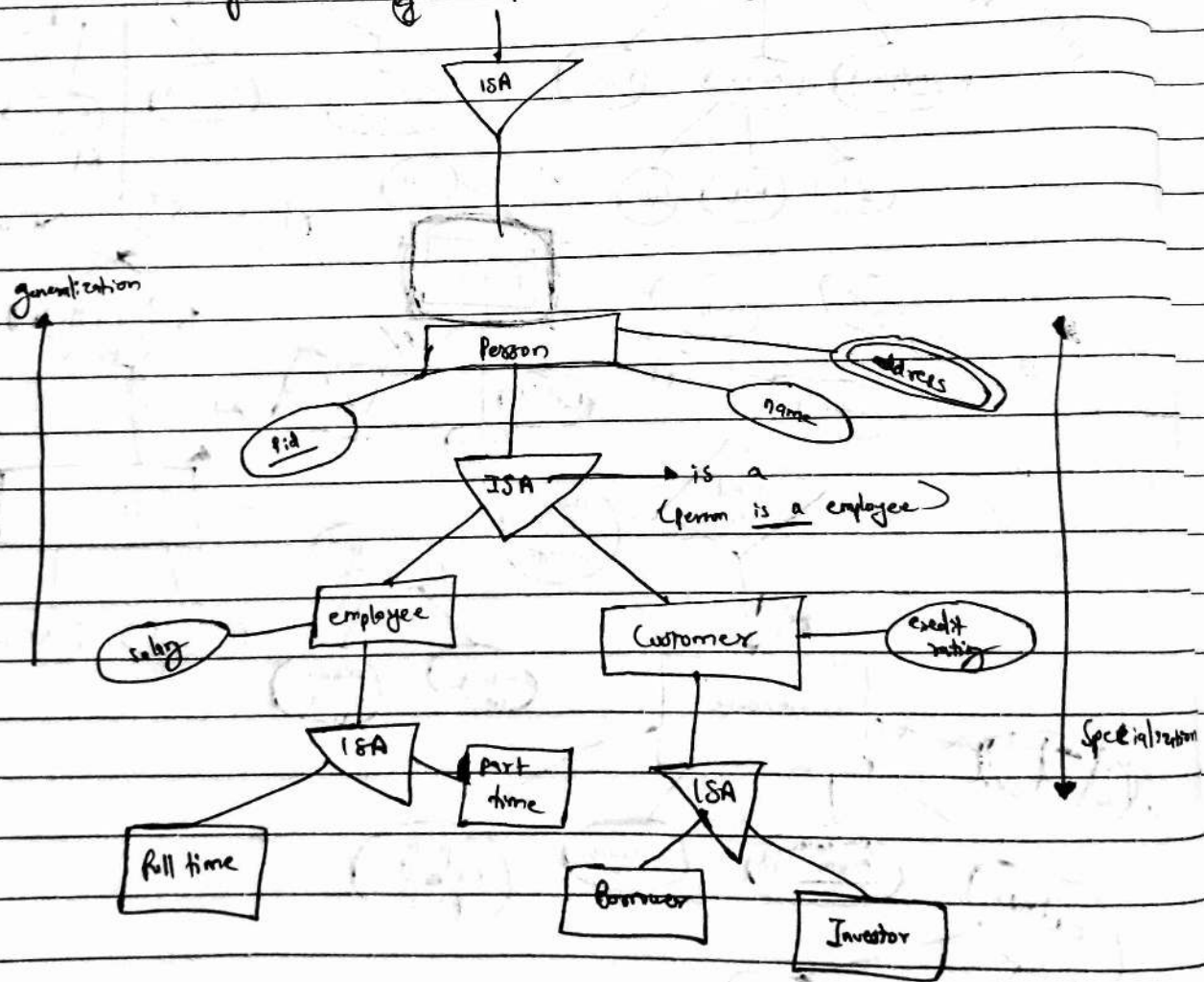
EER diagram
 enhanced or extended version of ER diagram

It provides more functionality than the ER diagram.

① Specialization → in this higher level entity sets are divided into lower level entity sets. (Top down)

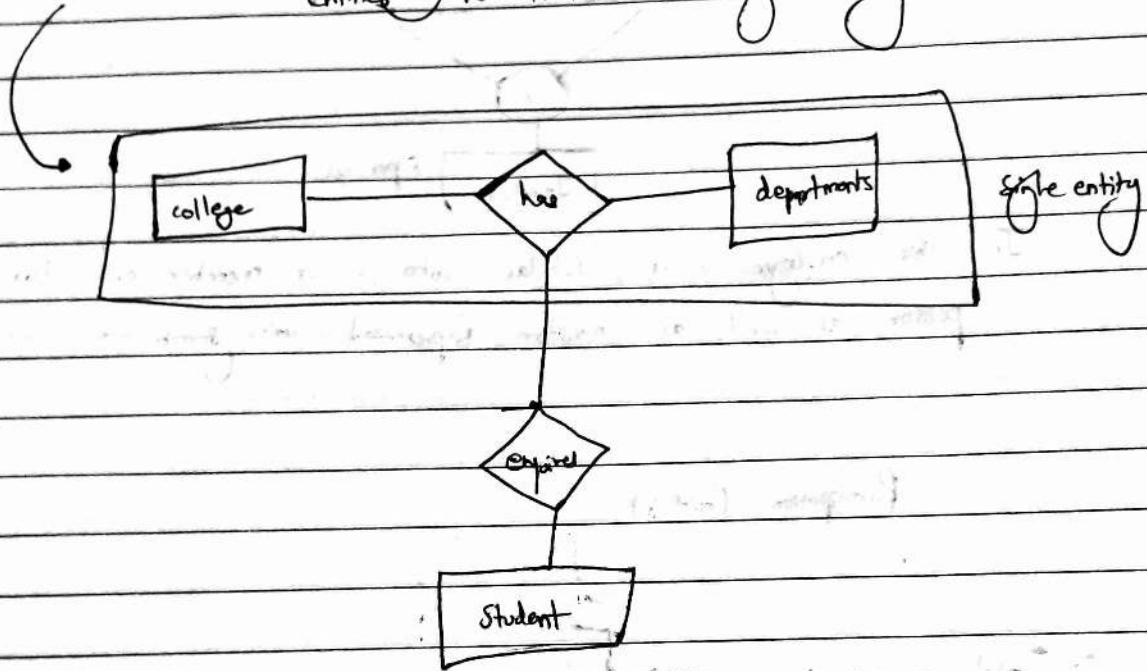
② Generalization → in this low level entity sets are merged into higher level entity sets. (Bottom up)

Symbol for both specialization and generalization :-



③ Inheritance → is the process in which all low level entities (sub-class) will inherit properties of higher level entity (super class).

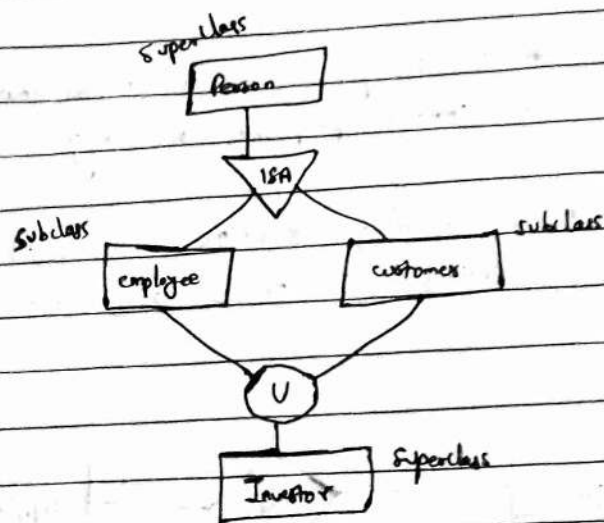
④ Aggregation → is modelling technique in which "rel" between two entities is treated as single entity.



In this diagram, "rel" between college and department acting as a single entity which is in "rel" with another entity called student. If student visits a college, he will enquire about both college as well as the department.

⑤ Categorization → is process in which subclass will have more than one superclass.

continued on
next page

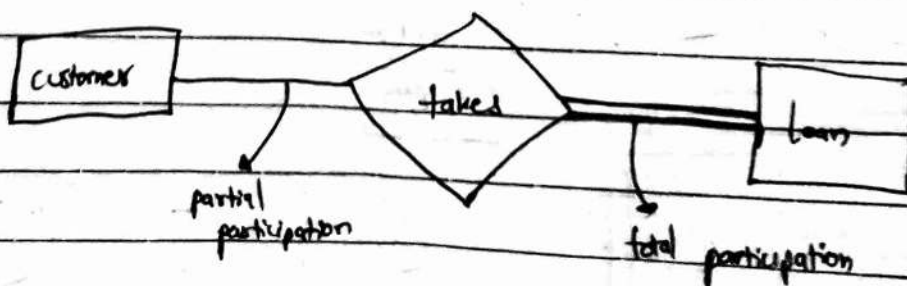


In this employee is a subclass who is a member of two superclasses: person as well as investor. Represented with symbol $\rightarrow \bigcirc U$

Participation Constraint

① Total participation \rightarrow If every entity from entity set E , is participating in relⁿ set R , then it is termed as total participation.

② Partial participation \rightarrow If some ~~some~~ of the entities are even/are participating in relⁿ set R , then it is termed as partial participation.



Bank is not an entity as it is an organization

PRG. NO.	
DATE	/ /

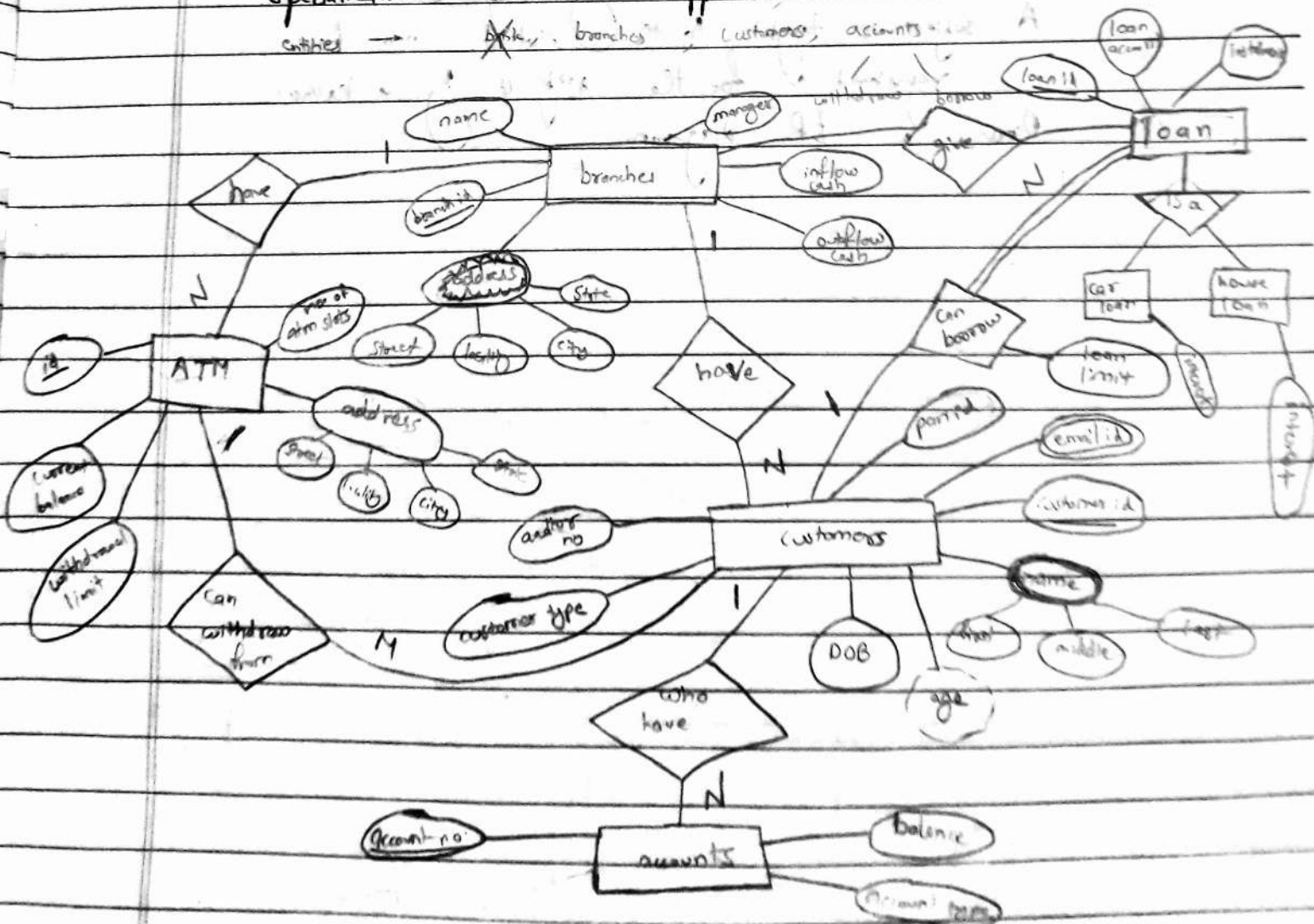
Not all customers are taking loan from a bank and therefore we have partial participation (for customers)
 For each ~~loan~~ given we have customer id associated. Therefore we have total participation (for loan)

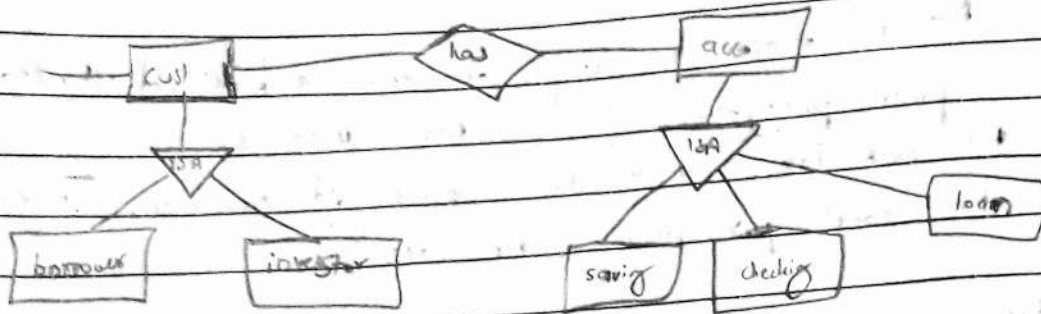
EER examples

- 1) Bank has many branches with many customers
- 2) A customer can open different kinds of accounts with the bank.
- 3) Any customer of the bank can take loan from the bank.
- 4) Banks have also installed ATM machines from which customer can withdraw money.

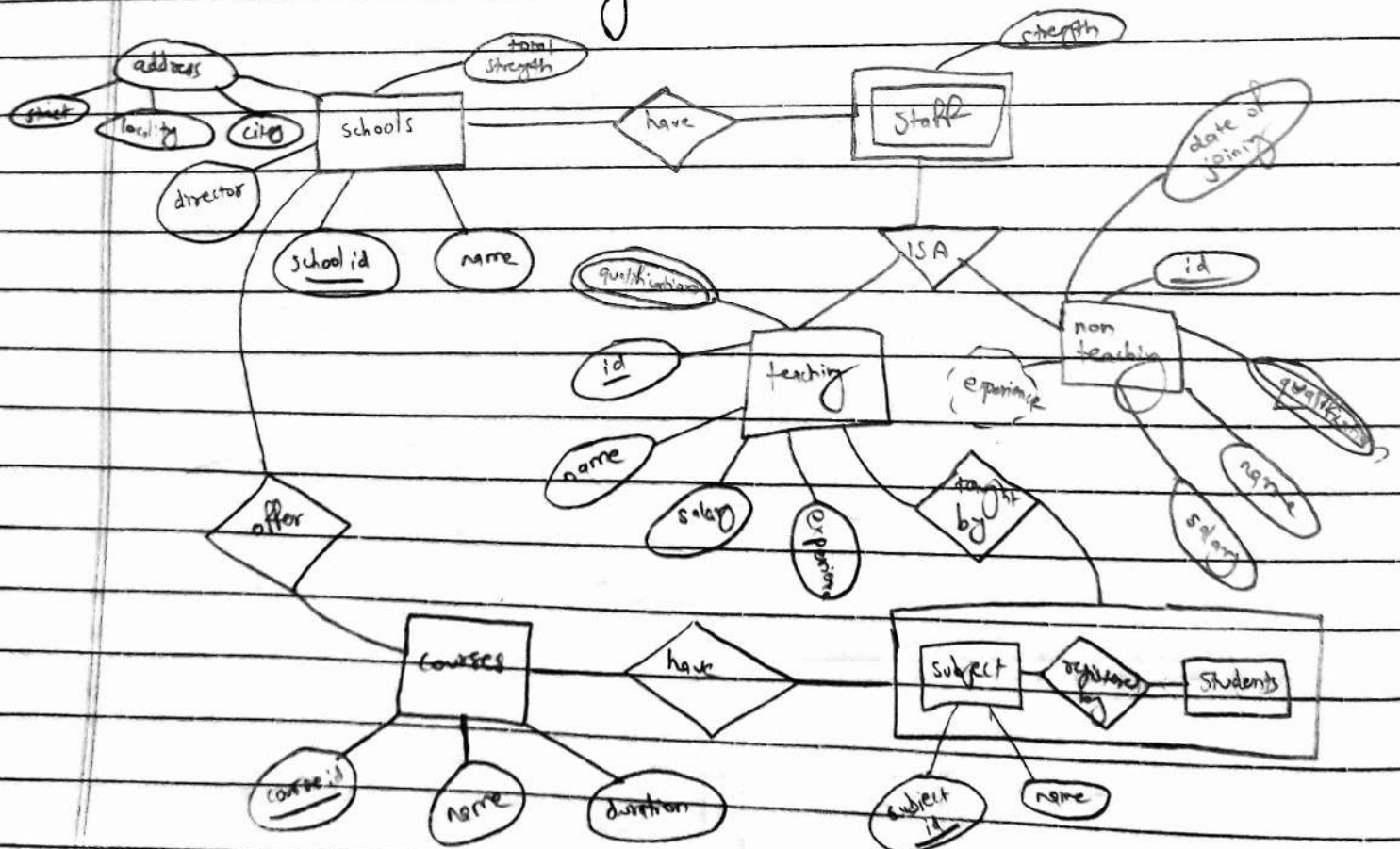
Draw EER diagram and use aggregation, generalization or specialization where ever applicable.

Entities → ~~bank~~, branches, customers, accounts





- Q. University has many academic units called schools.
 Each school is headed by a director of school.
 School has teaching and non teaching staff.
 School offers many courses.
 A course consists of many subjects.
 A subject is taught to the student who have
 registered for the subject by a teacher.
 Draw ER diagram



Mapping of ER/EER models into relational model

Rules for mapping

① Entity

① Strong → create separate table

② weak → separate table

← Primary key of main table +

Any ~~is~~ "alt." key of weak entity

② Attribute

① Composite attr. → no separate table

② Multivalued → create separate table

③ Relationship

① 1:1 → No separate table to be created.

② 1:N → No (1's primary key added in N's table)

③ M:1 → No

④ M:N → Yes

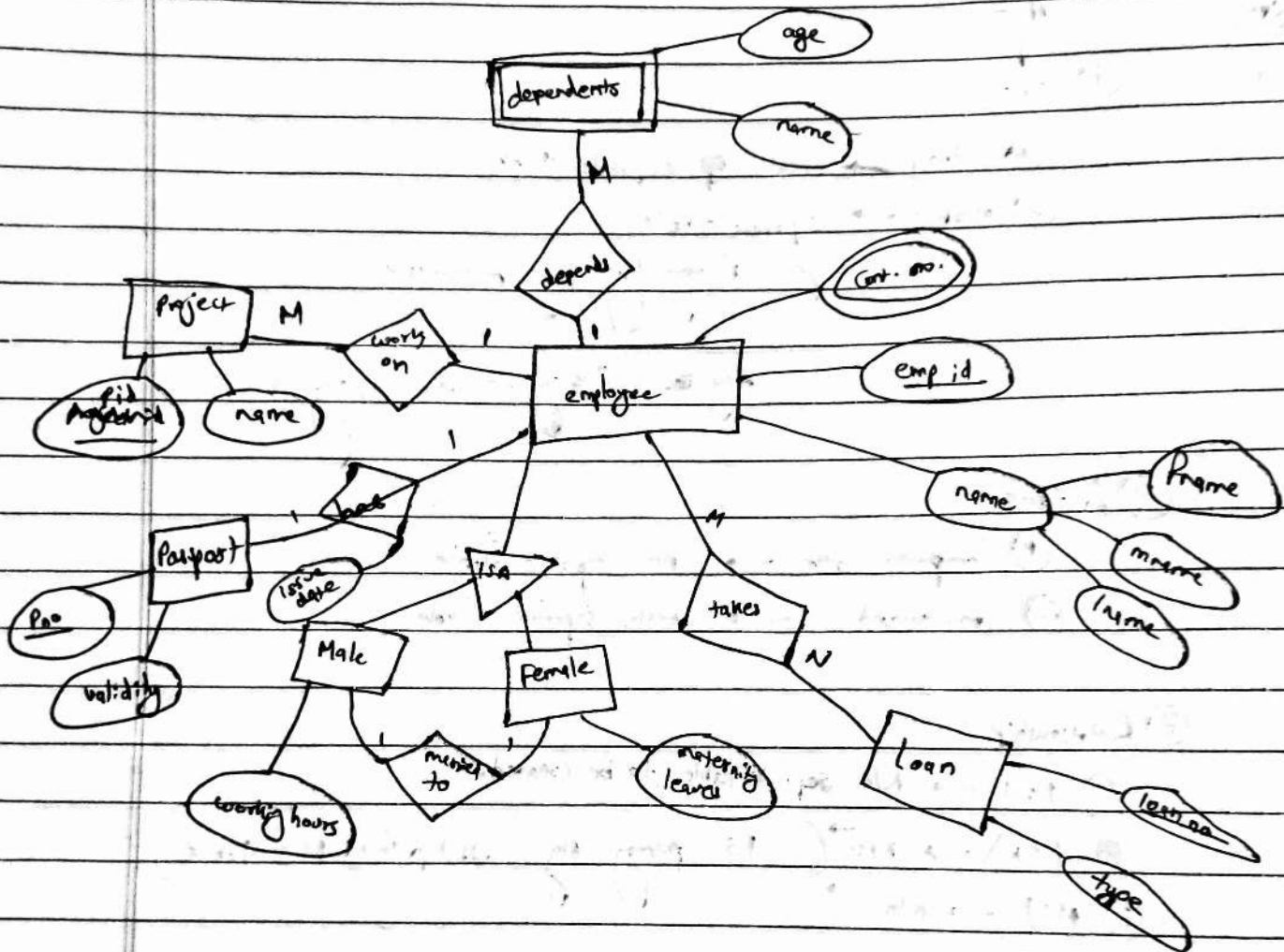
④ Inheritance

Each subclass will have primary key of parent class

⑤ If descriptive attr. is present, create separate table.

Descriptive attr. \rightarrow attr. for relⁿ

PAGE NO.	
DATE	/ /



① emp (empid, fname, mname, lname)

② contact (empid, contact no)

③ project (pid, name, empid)

④ mle (Mempid, working hours, fempid)

⑤ female (fempid, maternity leaves)

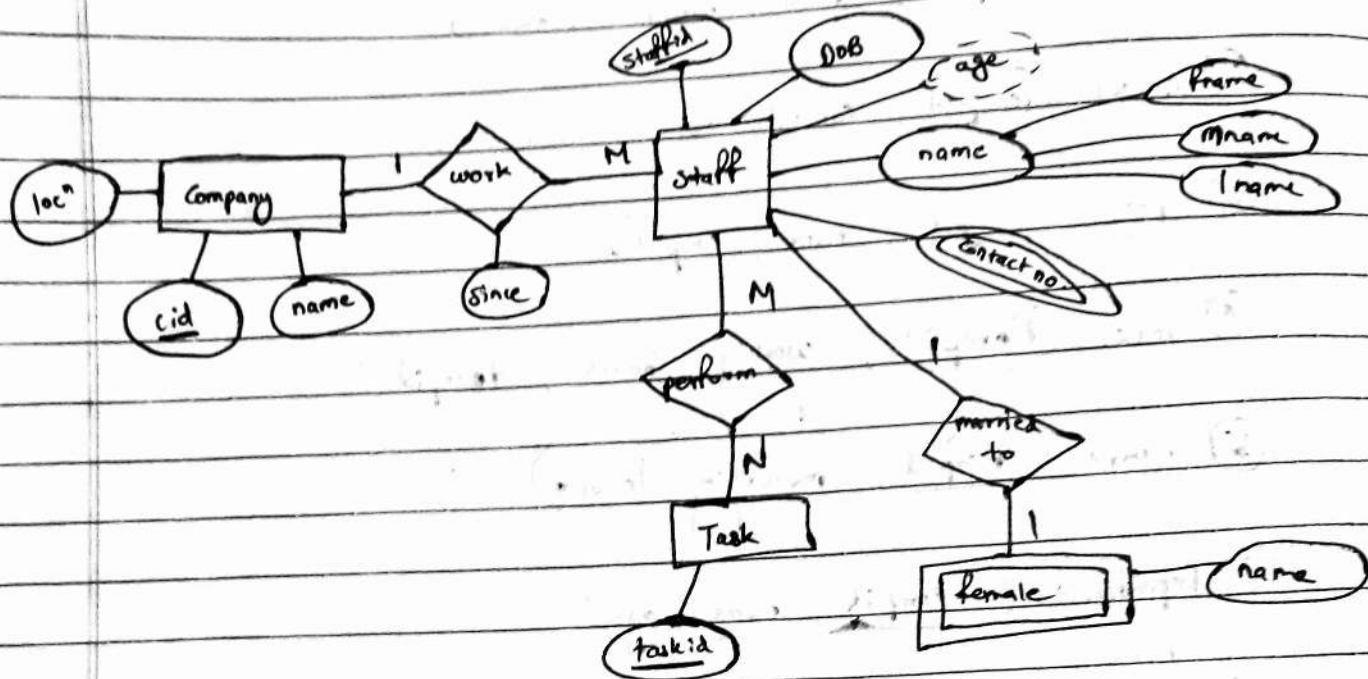
⑥ Dependencies (empid, name, age)

⑦ passport (pno, validity)

⑧ has (pno, empid, issue date)

⑨ loan (loanno., type)

⑩ takes (empid, loanno.)



Company (cid, loc, name)
 work (cid, staffid, since)

staff (staffid, DOB, ~~age~~, frame, mname, lname, ~~cid~~)
 contact no (staffid, contact no.)

task (taskid)

perform (taskid, staffid)

female (staffid, name)

Primary key of main entity +
 any attribute of weak entity

No need to create coln
 for age

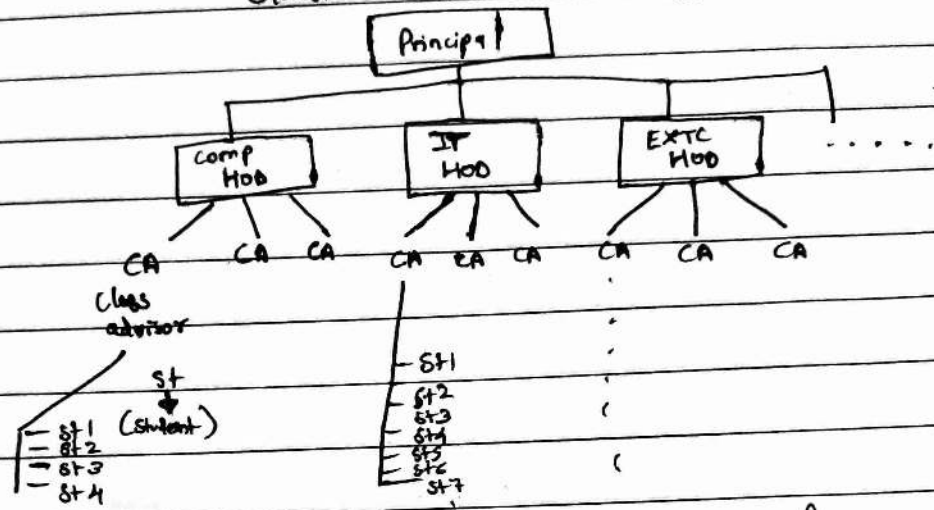
it will not
 come here
 as 'work'
 table has cid
 attribute

Data models in DBMS

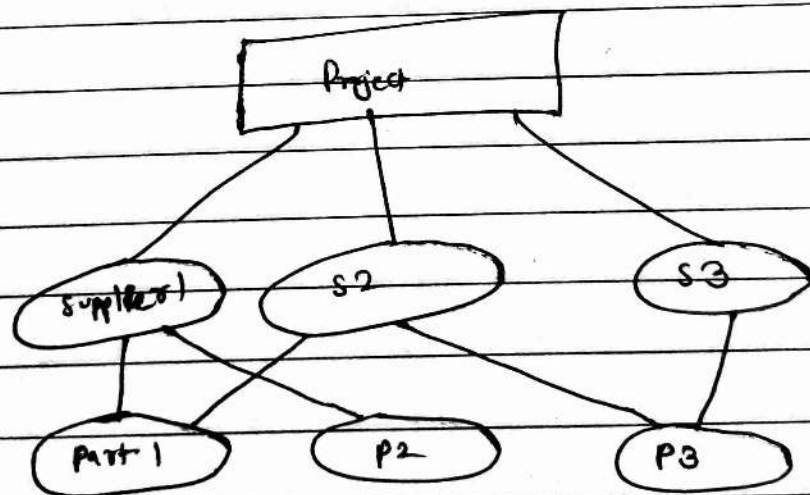
It describes how components are structured together.

① Hierarchical model: It represents hierarchy of an organization.
(level of detail)

It supports only 1:M cardinality.
Oldest model.



② Network model: It supports all the types of cardinalities.

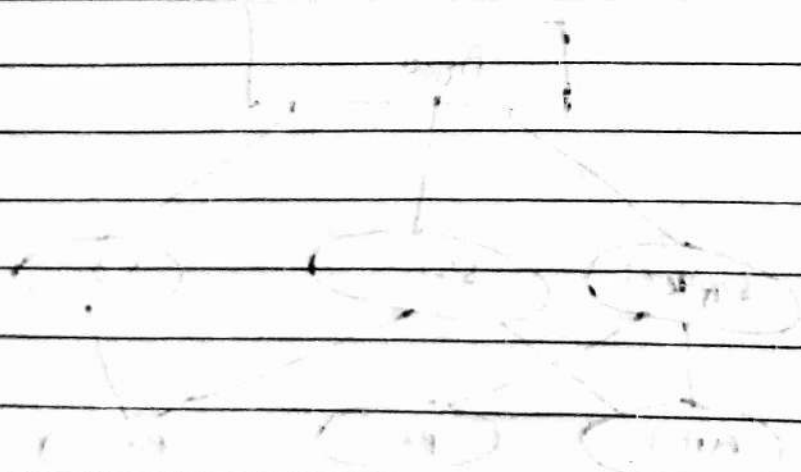
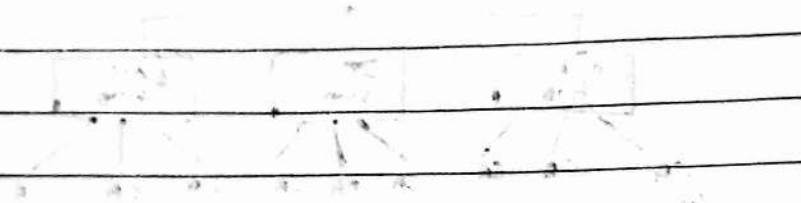


③ ER model: It represents entities and their relⁿ.

④ Relational model: If data is represented in the form of 2-D table it is relⁿ model.

Relational schema

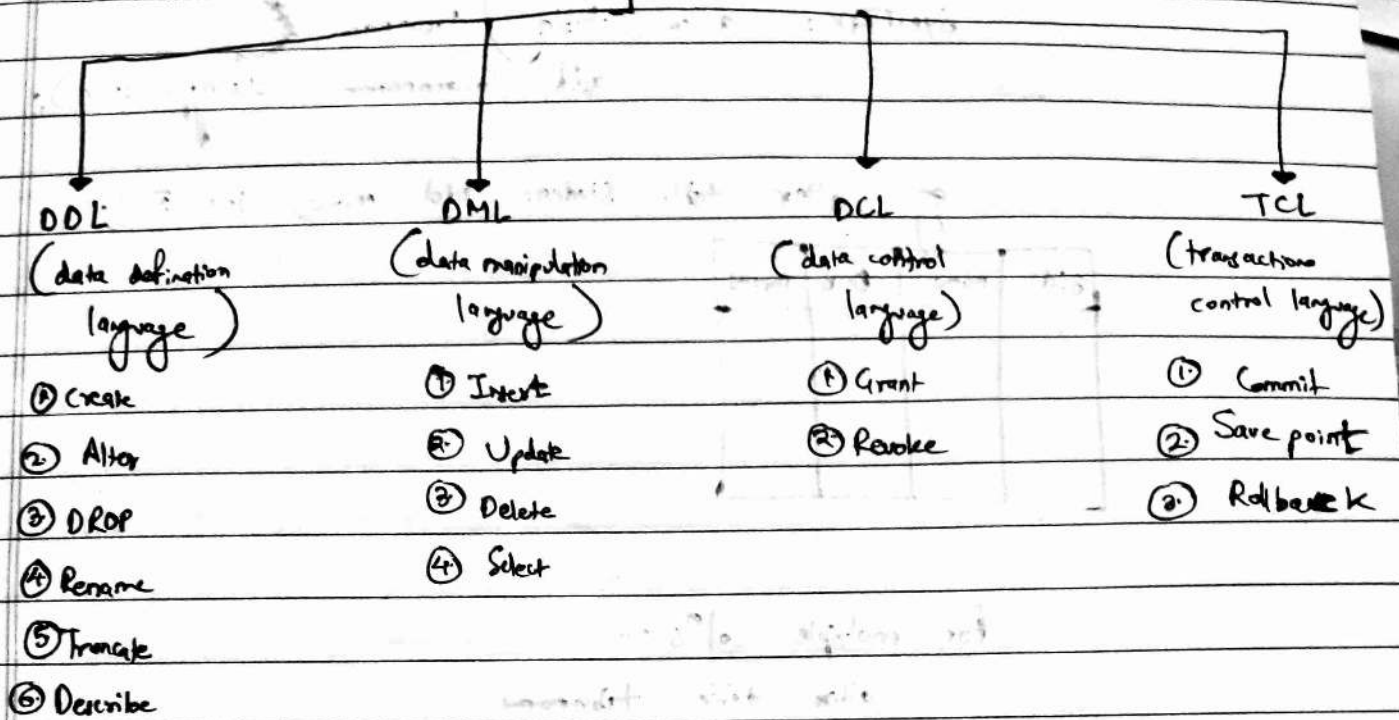
It is collection of metadata which describes conceptual/ logical schema.



SYNTAX for date → yyyy-mm-dd

PAGE NO.	
DATE	/ /

SQL (Structured Query language)



Data definition language

① Create : It is used to create new table for DB.

SYNTAX : Create table tablename (

columnName1 datatype(size),

columnName2 datatype(size),

.....) ;

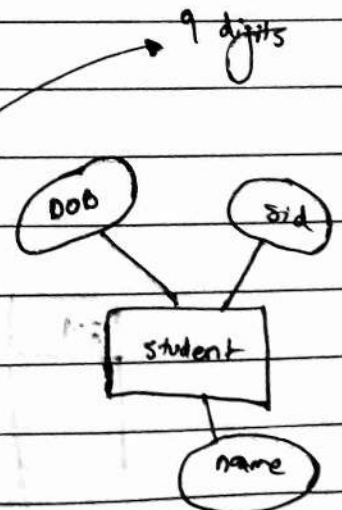
eg:- create table Student (

name sid int (9),

name varchar(20),

DOB date);

Sid	name	DOB



② Alter: It allows users to modify the structure of DB.

SYNTAX: alter table tablename
add columnname datatype(size);

eg:- alter table student add marks int(3);

sid	name	DOB	marks

For multiple col's :-

alter table tablename

add columnName1 datatype(size);

add columnName2 datatype(size);

For ~~adding~~ after a particular colⁿ and not at the end:-

alter table tablename

add ColumnName datatype(size)

after columnName;

sid	name	DOB	marks

eg:-

alter table student

add contactno int(10)

after name;

For altering at the first position of table :-

alter table tablename
add columnName datatype (size)
first;

emailId

Sid	name	DOB	mail

eg:- alter table student
add emailId varchar (20)
first;

To modify datatype of a column:-

alter table tablename
modify column
columnName datatype (size);

eg:- alter table tablename
modify column
sid varchar (5);

To change sid
from 513 to B-513

To drop a column from DB:-

alter table tablename
drop column
columnName;

alter table tablename
drop column
columnName1,
drop column
columnName2;

alter table student-
drop column
DOB;

Sid	name	DOB

drop

③ Rename → It allows you to rename the table name or columnName.

SYNTAX:- rename table
oldname to newname;

eg:- rename table
student to student_info

↓
will change the table ~~name~~ name to 'student_info'.

To change column name:-

SYNTAX:- alter table tableName

change column

oldvalue : newvalue datatype(size);

[there is no 'to' here.]

eg:- alter table student_info
change column
sid rollno;

④ DROP → used to drop the entire table

SYNTAX:- drop table tableName

eg:- drop table student_info;

In this attributes as well as records are deleted.

⑤ Truncate :- It deletes all the records from the table but attribute names will be retained.

SYNTAX: truncate table tablename;

eg:- truncate table student-info;

⑥ Describe :- It describes the structure of the table.

SYNTAX: desc tablename;

eg:- desc student-info;

(Will describe the structure along with columnName, its datatype and size.)

DML

(Data manipulation language)

① insert

② update

③ delete

④ select

① Insert → used to insert no. of rows into the table

SYNTAX:- insert into tablename

values (value1, value2, value3, ...);

insert into student values (1, 'Parth', '2000-1-1');

id	int(4)
name	varchar(16)
DOB	date

insert into student values (2, 'XYZ', NULL);

↳ no DOB found

② Update :-

Update command is used to modify existing value of column.

SYNTAX: update ~~table~~ ^{tablename} ~~table~~
 set newvalue
 where oldvalue;

eg. - update student set id = 3 where
 name = 'XYZ'

(age id = 3 where name = 'XYZ')

[if multiple values have the same value,
 both the places, value will be changed]

③ Delete :- Delete command is used to delete no. of rows from the table.

SYNTAX: delete from tablename
 where condition;

eg. - delete from student
 where id = 3;

(will delete rows where id = 3)

④ select :- select command is used to select no. of rows from table.

SYNTAX: $\left[\begin{array}{l} \text{select column name(s) from table name} \\ \text{where condition;} \end{array} \right.$
 (* \rightarrow all the column names are selected.)

select * from prof_details
where sal = 60000;

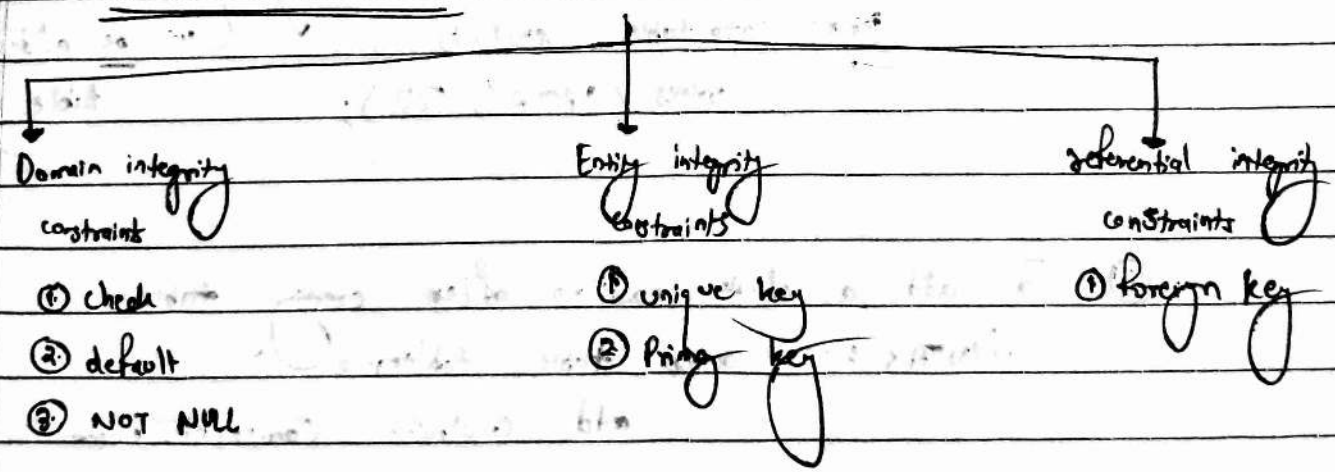
/ * All columns are selected */

(id, name, sal)
 select id from prof_details
(id) where sal = 60000;

/ * only id is displayed */

id	name	sal
1	XYZ	60000
2	POK	60000
3	ABC	65000

DBMS constraints



Constraints:- Before entering values in the table, values are checked against the constraint. If constraints are satisfied, then only values are inserted in a table.

for data integrity

① Check constraint

① SYNTAX :- create table tablename (
col1 datatype (size),
col2 datatype (size),
check (condition));

eg:- create table ~~tablename~~ ^{Students} (
name varchar (16),
age int (2),
check (age > 18));

interest
this becomes
the default
constraint name

insert into table Students
values ('ABC', 16); X (will not be added
in table)

insert into table students
values ('PQR', 22); ✓ (will be added in
table)

② To add a check condition after creating table

SYNTAX :- alter table tablename
add constraint constraint_name
check (condition);

eg:- alter table students
add constraint chk1
check (age > 18);

① To drop constraint from table

SYNTAX: alter table tablename

drop constraint constraint_name;

② default constraint

default constraint is used when we don't know value for a specified column.

③ Creating table with 'default':

SYNTAX: create table tablename (

c1 datatype (size),

c2 datatype (size),

c3 datatype (size) default 'value');

[Insert into student values (.....)]

[insert into table values (.....)] → without the default corresponding val. specified

④ Change default after creating table

SYNTAX: alter table tablename alter columnname set default 'value';

↓
alter table tablename

alter columnname

set default 'value';

⑤ drop a default value

SYNTAX: alter table tablename

alter column columnname

drop default;

→ We can have duplicate values but null values is not accepted.

③ Not NULL constraint

SYNTAX:- create table tablename (
 c1 datatype(size),
 c2 datatype(size),
 c3 datatype(size) not null);

④ after table created:-

SYNTAX:- alter table tablename
 modify column columnname datatype(size)
 not null;

→ create table student (
 id int(9),
 name varchar(25) not null,
 city varchar(50));

insert into student values (3, 'ABC'); ✓
 (3, NULL, 'MUM'); X
 (2, 'ABC', NULL); ✓
 (2, 'ABC', 'MUM'); ✓

⑤ drop 'not null' :-

alter table tablename
 modify column columnname datatype(size)
 null;

⑦ Primary key - it is the combination of unique + not null

SYNTAX :- create table tablename (
 c1 datatype (size), Primary key
 c2 datatype (size),
 c3 datatype (size));

insert into tablename
(1, 'A', 'Mum'), ✓
(2, 'B', 'Banglore'), ✓
(1, 'C', 'DEF'), X
(NULL, 'D', 'EFG'), X
(3, NULL, NULL), ✓

order of execution ↓

① ~~after~~ after table created

alter table tablename
add constraint constraintname
primary key (columnname);

② drop 'primary key' constraint from a constraintname/column
after table tablename
drop primary key;

Referential integrity constraint :- It focuses on primary - foreign key relationship.

sid	name	age	deptid	deptid	branch
1	A	18	10	10	IT
2	B	19	20	20	COMP
3	C	19	10	80	EXTC
4	D	21	20		
5	E	20	30		

Student

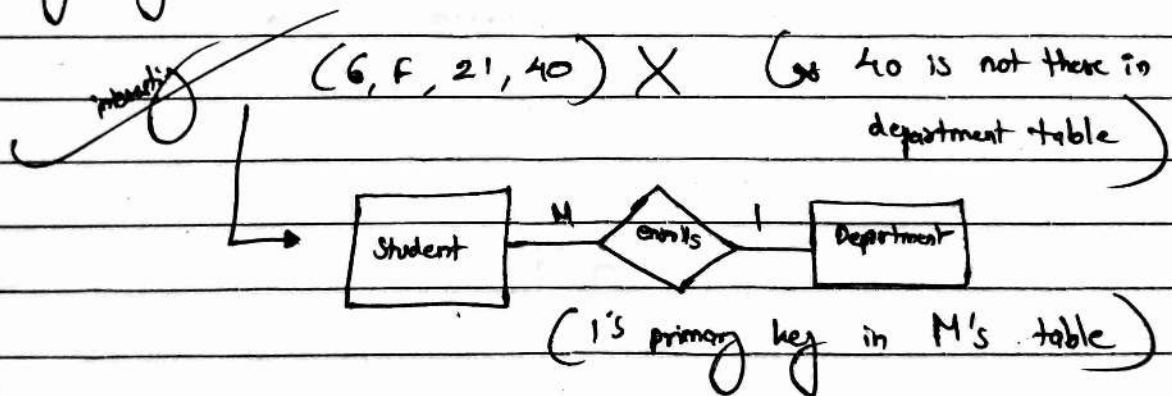
Department

sid → primary key for 'student' table

deptid → for 'department' table

also acts as a normal coln in student table, referred as foreign key.

→ foreign key will have NULL as well as duplicate values.



SYNTAX :- create table tablename (
 c1 datatype (size),
 ...
 cn datatype (size),
 constraint constraintName foreign key (columnName)
 references another table name (columnName),

eg:- create table student (
 ...
 constraint fk1 foreignkey (deptid)
 references dept (deptid);

① after table created

SYNTAX:- alter table tablename
add constraint constraintname foreignkey (columnname)
references another table name (columnname);

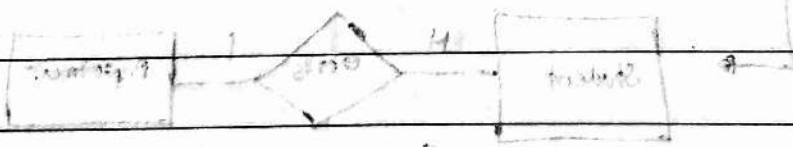
② drop foreignkey from a 'constraint name'

① alter table tablename

drop index constraintname;

① alter table tablename drop foreign key constraint name;

space



Transaction control language

It is used in client-server architecture

Transaction → series of statements used in client-server architecture.

- ① Commit
- ② Roll back
- ③ save point

Commit → used to save changes permanently in the DB.
once changes are made, changes cannot be undone.

SYNTAX :- Commit;

rollback → used to restore the old values upto last committed step.

SYNTAX: rollback;

savepoint → used to save changes temporarily in DB

Creating of savepoint: -

SYNTAX :- savepoint savepoint_name;

roll back with savepoint: -

SYNTAX: - rollback to savepoint_name;

eg:-

id	name	sal
1	Ajay	9000
2	Vijay	5000
3	Sanjay	6000

prf

① begin;

② update prf set sal = sal + 1000 where id = 1;

③ select * from prf; (will show ajay → 10000)

④ rollback;

⑤ select * from prf; (will show ajay → 9000)

⑥ Same as ②

⑦ commit; (changes made permanent)

⑧ select * from prf; (10k)

⑨ rollback;

⑩ select * from prf; (still shows 10k)

interesting

commit indicates client-server arch. has completed

PAGE NO.	
DATE	10/10/2020

use begin command again

- (11) begin; (as)
- (12) same as (2) (11k)
- (13) same as (2) (12k)
- (14) rollback;
- (15) select * from prof (10k)
- (16) start transaction; (to use savepoints)

- (17) savepoint sp1;
- (18) delete from prof where id=1; (agay X)
- (19) select * from prof;
- (20) savepoint sp2;
- (21) delete from prof where id=2; (vijay X)
- (22) select * from prof;
- (23) savepoint sp3;
- (24) delete from prof where id=3; (sanjay X)
- (25) same as (22)
- (26) roll back to sp2; (sanjay, vijay added)
- (27) same as (22)

savepoint 2 [by default savepoint 3 is performed]

id	name	age
0001	AGAY	1
0002	VIJAY	2
0003	SANJAY	3

SQL operators

Operator is a special keyword used in databases for describing operations. There are 6 types of operators in SQL:-

- ① Arithmetic
- ② Comparison
- ③ Logic
- ④ Set operators
- ⑤ Range searching
- ⑥ Pattern matching

① Arithmetic operators

operator	description	eg
$+$ (addition)	adds value in either side of operators.	$x + y = 30$
$-$ (subtraction)	Subtracts right hand operand from left hand operand.	$x - y = 10$
$*$ (mult.)	multiplies value in either side of operators.	$x * y = 200$
$/$ (divide)	left hand operand gets divided by right hand operand.	$x / y = 2$

$$x = 20$$
$$y = 10$$

what happens when we interchange
select id, sal
instead of select sal, id

PAGE NO.
 DATE
 TIME

check?

examples of arithmetic operators

empid	fname	lname	city	sal
1	Ajay	Sharma	Mumbai	5000
2	Vijay	Mehta	Pune	10000
3	Satyaj	Patil	Mumbai	7000
4	Jayesh	Sharma	Pune	2000

emp-table

Display sal, empid of the employees by adding 1000 to it.

select id, sal as old_sal,
sal + 1000 as new_sal from emp-table;

id	old_sal	new_sal
1	5000	6000
2	10000	11000
3	7000	8000
4	2000	3000

Display first name and sal, by dividing ~~from~~ 2.

select fname, sal as old-sal,
sal/2 as new-sal from emp-table;

fname	old-sal	new-sal
Ajay	5000	2500
Vijay	10000	5000
Sanjay	7000	3500
Jayesh	2000	1000

Display all the details of the employees by giving 5% raise in salary

Select empid, fname, lname, city, sal as old-sal,
 $sal + (sal * 5) / 100$ as new-sal from emp-table.

empid	fname	lname	city	sal
1	Ajay	Sharma	Mum.	5000
2	Vijay	Mahar	Pune	10500
3	Sanjay	Patil	Mum.	7350
4	Jayesh	Sharma	Pune	2100

50x5
=250

Comparison operator

① $=$ (equal)

Checks value of two operands:-

If equal \rightarrow returns true

② $!=$ (not equal to)

Checks value of two operands:-

Not equal \rightarrow returns true

Equal \rightarrow returns false

③ $<$ (less than)

Checks if value of left operand is less than value of right operand.

If yes \rightarrow returns true

④ $<=$ (less than, or equal to)

⑤ $>$ (greater than)

Value of left operand is greater than value of right operand.

If yes \rightarrow true

$>=$ (greater than or equal to)

① emp whose sal is 7000.

② display all the employees whose sal $>=$ 7000

③ display record of all the employees by adding 2000 to the salary only when sal < 7000 .

① select * from emp-table where sal = 7000.

② select * from emp-table where sal > 7000.

③ select emp-id, fname, sal as old-sal, sal + 2000 as new-sal where sal < 7000;

3 Sajay Patil Mumbai 7000

2 Vijay Mehra Pune 10000

3 Sajay Patil Mumbai 7000

emp-id, fname, lname, city, old-sal, new-sal

Logical operators

AND :- it returns True if all the conditions are satisfied.

OR :- it returns True if one of the conditions is satisfied

Q. Display the information of all the employees with last name as 'sanjay' and last name as 'patil'.

select * from emp-table where name = 'sanjay' and name = 'patil';

for or → [3, 5, 6]

emp-id	name	city	sal
1	ajay	sharma	5000
2	vijay	Melita	10000
3	sanjay	patil	7000
4	jyesh	Sharma	2000
5	mangesh	patil	15000
6	ajay	patil	7500

name name

Q. 'sanjay' or 'patil'

select * from emp-table where name = 'sanjay' or name = 'patil';

output → 3 —
5 —
6 —

Q. Name = 'gaj' or Name = 'sharma'

[1, 3, 4]

emp id	name	city
1	gaj	mumbai
3	sharma	mumbai
4	gaj	pune

Q. List all employees having salary ≤ 7000 and city as mumbai.

select * from employees where sal ≤ 7000 and city = 'mumbai';

1. gaj	mumbai	5000
3. sharma	mumbai	7000

Q. Display information of all employees having Name as 'gaj' or 'sharma' and last name as sharma

select * from employees where (Name = 'gaj' or Name = 'sharma') and
| Name = 'sharma';

1. gaj	mumbai	5000
4. sharma	pune	2000

Range searching operators

BETWEEN / NOT BETWEEN

Two types of operators are included in range searching: **IN** / **NOT IN**
 It is used to select the records from the table when we know the exact value

IN:- SYNTAX:- select column names (*) from table name *
 where column name in (value1, value2, ... valuen);

id	fname	lname	sal	city
1	ajay	sharma	5000	mum
2	vijay	mehta	6000	pune
3	sanjay	patil	7000	mum
4	jayesh	sharma	2000	pune

① select * from emp table where lname in ('sharma', 'patil');
1, 3, 4

② select * from emp table where (lname = 'sharma' and lname = 'patil');
 ↓
 { } empty set

neither
 lname not equal to 'sharma' or 'patil'
 nor equal to
 AND

or → same as ①

select * from emp table where (lname != 'sharma' and lname != 'patil');

2

Substitute is not in

2. ~~not~~ mehta 10000 pune

not equal to 'sharma' or 'patil'

↳ or

select * from emp table where (name != 'sharma' or name != 'patil');

1, 2, 3, 4

BETWEEN

↳ used to select values based on ranges

SYNTAX: select columnname (*) from tablename
where columnname between
min_value and max_value;

in, =, or

not in: !=, and

between: >, <

and,

<=

not between: <, >

or,

>

① select * from emp id where sal between 2000 and 7000;

1, 3, 4

NOT BETWEEN

① select * from emp id where sal < 2000 and sal > 7000

↳ { } empty set

② select * from emp id where sal < 2000 or sal > 7000

Not between

2

~~interesting~~ ① select * from emp where frame between 'Ajay' and 'Sanjay';
 as frame > 'Ajay' and frame ≤ 'Sanjay'

② select * from emp where frame between 'A' and 'S';
 as frame > 'S' and frame ≤ 'A'

~~very interesting~~ as 'Sanjay' > 'S',
 so we will discard Sanjay,
 only 'Ajay' and 'Jagdish'.

Set operators :- used to combine result set of

eid	cname	c city	c country
1	Pooja	Mum	Ind
2	Vijay	Delhi	Ind
3	Vijay	Berlin	Germany

Customer

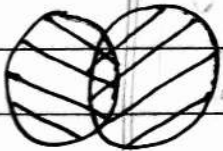
eid	cname	c city	c country
1	Kajal	London	UK
2	Vijay	Berlin	Germany
3	Sangam	NY	USA
4	Vishal	Texas	USA
5	Vijay	Delhi	Ind

Employee

used to combine output of multiple select statements. Four types:-

- union all
- union
- intersect
- minus

Union all :- It is used to combine two select statements together with keeping duplicate records.



SYNTAX :- select col1 from table

union all

select col2 from another table;

eg. - select cid from Customer

union all

select eid from employee

cid
1
2
3
1
2
3
4
5

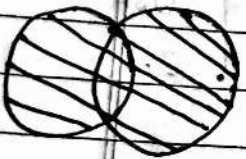
select eid from customer
union all
select eid, ename from employee

EXERCISE NO.	
DATE	/ /

limitation

In set operator ^① we should have equal no. of columns
and ^② their datatype should be matched.

Union: — It is variation from union all in which
duplicate values are removed and only distinct
values are displayed.



eg. select eid from customer

union
select eid from employee;

cid
1
2
3
4
5

name, city	name	city	Union
select name from customer	Prigunika	Mum	India
union	Vipul	Delhi	Germany
select ename, city from employee	Vijay	Berlin	UK
	Kalpene	London	USA
	Sanjay	NY	
	Vashid	Texas	
	Vijay	Delhi	

in this case
we compare (Vijay, berlin)
with
(Vijay, delhi)

intersect :- it selects common records from both the tables.

select ename, city from customer

intersect

select ename, city from employee



ename	city
Vijay	Reston

minus :- it will select uncommon records of first select statement.

select ename, city from customer

minus

select ename, city from employee



ename	city
Priyanka	Mum
Vipul	Delhi

③ select all employees whose name is 'ja' in it.

select * from emp
where name like '%ja%';

id	name	last name	sal
1	Vijay	Mehra	6000
2	Ajay	Chen	5000
3	Sanjay	Patil	7000
4	Jayesh	Sharma	2000

④ select all employees whose second of first name is 'a'.

select * from emp
where name like '_a%';

Sanjay
Jayesh

⑤ select all employees whose first name has exactly six characters.

select * from emp
where name like '____';

Sanjay
Jayesh

⑥ find all employees whose name should contain 'ja' but does not start with 'ja'.

⑦ select all the employees in which first name 'a' is in second position and name must end with 'y' or 'h'.

select * from emp
where name like '%ja%' and
name not like 'ja%';

o/p → 123

more efficient
select * from emp
where name like '_a%y' or name like '_a%h';
select * from emp
where name like '%ja%';
o/p → 3,4

Post this page

date

PAGE NO.	
DATE	/ /

Pattern matching operators

id	fname	lname	sal
1	Vijay	Mehra	10000
2	Ajay	Sharma	5000
3	Sanjay	Patil	7000
4	Jayesh	Sharma	2000

emp.

① select all the employees whose fname ends with 'y'.

In pattern matching we have:-

- like / not like
- is null / is not null

like: SYNTAX: select columnName (*) from tablename
where columnName like 'condition';

wildcards

L S — → exactly one value
% → 0 or more characters

select * from emp
where fname like '%y';

Vijay
Ajay
Sanjay

② select all employees whose last name starts with 'S'

select * from emp
where lname like 'S%';

Ajay Sharma
Jayesh Sharma

is null / is not null

↓
value is Null, value is not null

id	fname	lname	sal
1			
2			
3			
4			
5			

We are adding val.
5 | Satish | Bhare | NULL
to the existing table

① select * from emp where sal is null;

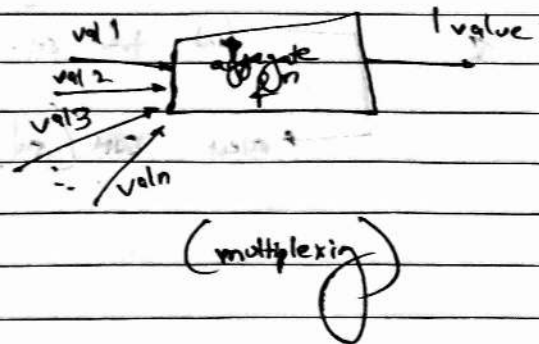
id	fname	lname	sal
5	Satish	Bhare	NULL

② select * from emp where sal is not null
↳ 1 to 4

grouping, ordering clause and any more fns

Aggregate Functions

id	fname	lname	sal
1	Ramesh	Kale	6400
2	Vijay	Sharma	9400
3	Sanjay	Patil	67000
4	Mukesh	Patil	4200
5	Ajay	Blue	10000
6	Mayraj	Bhasale	15600
7	Ramod	Patil	NULL
8	NULL	NULL	NULL



SYNTAX: — select aggregate fn name (column name) from tablename where condition;

Aggregate fn will take set of values as an input and produce only a single value as an output.

SUM (colname) → provides total of selected column excluding NULL values.

AVG (colname) → gives avg. excluding NULL

MIN (colname) → displays minimum value of a specified col.

MAX (colname) → displays max value for a specified col.

COUNT (colname) → gives total no. of values of a particular column excluding NULL.

~~including~~ includes NULL val

COUNT (*) → displays no. of records, a particular table has.

Q. find total sal. of employee :-
→ select SUM(sal) from emp;

sum(sal)
112000

select SUM(sal) as Total_salary from emp;

not SUM will exclude entire row if for SUM(sal+id) either sum OR id is NULL.

select SUM(sal+id) from emp; → sum 7, 8 excluded as salary is NULL.

sum(sal+id)
112021 → (1 to 6)

Q. avg salary of employees
select AVG(salary) from emp;

avg(sal)
18666.67

Q. find avg salary of employees where salary is greater than 12000.

select AVG(salary) from emp where salary > 12000;

avg(sal)
21000

Q. find highest salary of employee

select MAX(sal) from emp; → 67000

Q. find smallest salary of employee

select MIN(sal) from emp; → 1200

Q. display total no. of records in employee table has

select COUNT(*) from emp; → 8

Q. select COUNT(sal) from emp;

→ 6 (because COUNT(column) does not accept NULL)

Q. no. of employees whose name starts with 'p'

select COUNT(lname) from emp where lname like 'p%';

Q. no. of NULL records in sal column has

~~select COUNT(sal) from emp where sal is null;~~ (does not run in MySQL)

~~select COUNT(sal) from emp where sal = Null;~~ (does not run in MySQL)

~~select COUNT(sal) from emp where sal not like ' %';~~

~~select COUNT(*) - select(sal) from emp;~~ → 2

Q. total no. of values, total null values, not null values for sal column.

select count(*), count(*) - count(sal), count(sal) from emp;

count(*)	count(*) - count(sal)	count(sal)
8	2	6

Not NULL values in table

Group by

Set of values are combined together and on these set of values aggregate fn is applied.

SYNTAX:- select columnName, aggregate_fn_name (columnName) from tablename where condition group by (columnName), (columnName2);

display total salary of employee according to the department

select department, sum(sal) from emp group by deptname;

Table (in the behind table we add deptname)

id	deptname	dept	sum(sal)
1	Sales	Sales	
2	acc	acc	24400
3	sales	HR	4200
4	HR	marketing	10000
5	marketing	Sales	73400
6	acc		
7	HR		
8	HR		

because according to deptname we have to display salary

display total salary of employees whose sal > 9000

9 | marketing | 10000 | sales

select deptname, sum(sal) from emp where sal > 9000 group by deptname;

dept.	sum(sal)
acc	24400
marketing	10000
Sales	67000

select total salary of employee according to deptname and frame
select frame, deptname, sum(sal) from emp group by deptname, frame;

exp 6. To study and implement different types of databases in

mysql -u root -p

-u → username

-P → password

show databases; → for displaying how many databases are currently in the system.

create database b513;

use b513

→ use our db

show tables; → to show our list of tables

exp.4 To study and implement types of constraints in MySQL

exp.5 To study and implement TCL commands

deptname	Empname	sum(sal)
acc	manoj	15000
acc	vijay	9400
HR	null	null
HR	mukesh	4200
HR	pramod	null
marketing	gjay	10000
sales	rimesh	7400 → (6400 + 1000)
sales	sanjay	67000

Q. display total salary of emp according to deptname where total sal is greater than 10000

select deptname, sum(sal) from emp where sum(sal) > 10000 group by deptname;

interesting

(NOT SUPPORTED)

WHERE clause is not applicable for aggregate fⁿ.

∴ they introduced having clause

select deptname, sum(sal) from emp where

group by deptname having sum(sal) > 10000;

dept	sum(sal)
acc	24400
sales	73400

Aim: To study basic concepts of DBMS

- defⁿ of DB and DBMS
- difference between filesystem and DBMS
- arch. of DBMS
- diff. tools available in market related to DBMS
- characteristics of DBMS

→ add fname and lname together

→ select concat(fname, lname) from emp;

AJAY sharma
VIJAY patil
MUKESH Patel

add fname and lname with separator

→ select concat_ws('-', fname, lname) from emp;

AJAY-sharma
...

after table emp add column combine;

update emp set combine = concat_ws('-', fname, lname);

→ Create a new col and insert into the column values.

Having clause: — SYNTAX: —

select column name
 aggregate - fn - name (column name)
 from tablename where
 condition group by
 column name 1, column name 2
 having
 aggregate - fn - name (column name)
 condition;

Q. display total salary of emp according to deptname where total sal is greater than 10000.
 and sal is greater than 9000.

dept.	sum(sal)
acc	244000
sales	67000

select deptname, sum(sal) from emp where sal > 9000 group by deptname
 having sum(sal) > 10000

String manipulation

- ① upper (cn) — converts string into uppercase
- ② lower (cn) — " " lowercase
- ③ length (cn) — returns length of string
- ④ replace (cn, 'old value', 'new value') — replaces old value with a new value in a string
- ⑤ concat (first-col-name, second-col-name) — add two strings together
- ⑥ concat_ws ('separator', fcn, scn) — add two strings with separator

emp-id	fname	lname	sal	city
1	AJAY	SHARMA	2000	Pune
2	VIJAY	PATIL	5000	Mum
3	MUKESH	PATEL	2500	Kol

Q. display fname of emp in small letters and lname of emp in capital letters
 select upper(fname), lower(lname) from emp;

Q. replace each 'A' of first name by 'AA'
 select replace(fname, 'A', 'AA') from emp;

AAJAY
 VAAJAY
 MUKESH

→ nested string fns.

NOTE: — we can use lower(replace(---));

Q. display length of fname column
 select fname, length(fname) from emp;

fname	length(fname)
AJAY	4
VIJAY	5
MUKESH	6

Joins in DBMS

Joins are used to combine output from two different tables.

emp		orders		
eid	fname	pid	pname	eid
04	Vijay	234	shirt	7369
7369	Sanjay	657	top	7521
7419	Mukesh	865	tshirt	7521
7521	Pranod	878	Jeans	NULL

Q. display name of the employees who have ordered products

select emp.fname, orders.pname from emp, orders
where emp.eid = orders.eid;

fname	pname
Sanjay	shirt
Pranod	top
Pranod	tshirt

Q. select names of employees who have purchased shirt

select emp.fname, ^{orders.pname} from emp, orders where emp.eid = orders.eid
and orders.pname = 'shirt';

fname	pname
Sanjay	shirt

Types of joins :-

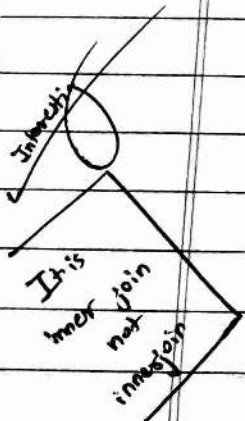
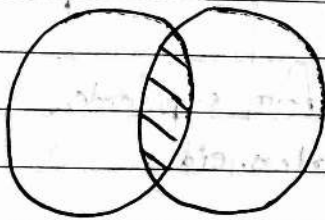
- ① Inner join
- ② Equi join
- ③ Natural join
- ④ Left join
- ⑤ Right join
- ⑥ Full join
- ⑦ cross/cartesian join

① Inner join - it selects rows from both tables when specified condition is specified. (CAN HAVE ANY TYPE OF OPERATOR PRESENT IN 'ON' CONDITION)

SYNTAX:

select enames from first table name
inner join second table name
on

first table.keyfield > second table.keyfield
where condition;



Q. display name of the emp who have ordered products.

select emp.name, orders.pname from emp

inner join orders products

on

emp.eid = orders.products.eid;

Q. with order as shirt

where orders.pname = 'shirt';

② Equi join:- If inner join condition contains equality operator '=', then this type of a join is equi join.

eg:- this is equi join

(In Equi join, we don't have where condition.)

Definition

③ Natural join:- Variation of inner join, where no 'ON' condition is present. Selection is done based on matching columns. This selection is performed automatically by DBMS.

SYNTAX:- select cnames from firsttablename
natural join secondtablename;

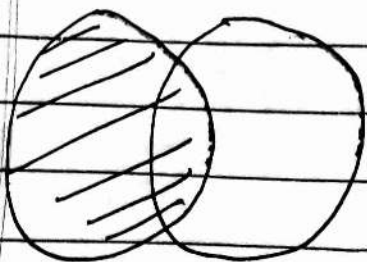
(ON "firsttable.keyfield = secondtable.foreignkeyfield")
if this is not necessary

④ Left join:- it selects rows from left table and if no value is present in right table, NULL is returned.

SYNTAX:- select cnames from firsttablename

left join secondtablename
on

firsttable.keyfield = secondtable.foreignkeyfield

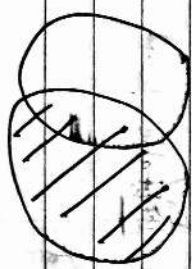


Q. (Same as kind)

select emp.id, emp.name, orders.pid, orders.pname
from emp
left join orders
on
emp.id = emp.orders.id;

emp.id	emp.name	orders.pid	orders.pname
04	Vijay	NULL	NULL
7389	Samyak	234	Shri
7499	Nikhil	NULL	NULL
7581	Ramod	657	top
7581	Ramod	865	thrt

5. Right join :- It selects rows from right table and if no matching is found, NULL is written



SYNTAX:-

select cols from firsttable
right join secondtable
on

firsttable.name, keyfield =
secondtable.name, keyfield

Q. select emp.eid, emp.fname, orders.pid, orders.pname
 from emp
 right join orders
 on emp.id = orders.id;

emp_eid	fname	pid	pname
7869	Sanjay	234	Shirt
7521	Pranod	657	Top
7521	Pranod	865	to hirt
NULL	NULL	878	jeans

Full join:— It returns rows from both tables and if no match is found, NULL is returned.
 on firsttablename.keyfield = secondtablename.keyfield

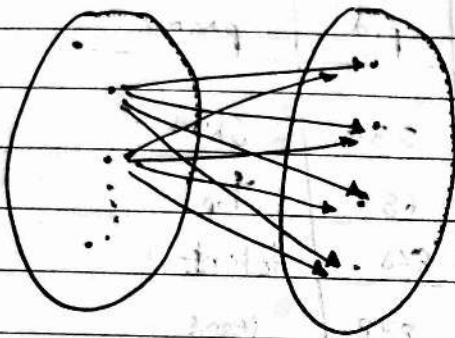
SYNTAX:— select cols from firsttable - name left join secondtablename
 where union
 select cols from firsttablename right join secondtablename
 on
 firsttablename.keyfield = secondtablename.foreignkeyfield;

Q.

eid	fname	pid	pname
04	Vijay	NULL	NULL
7869	Sanjay	234	shirt
7197	Ramkesh	NULL	NULL
7521	Pranod	657	top
7521	Pranod	865	shirt
NULL	NULL	878	jeans

⑦ Cross join :- each part of left table is matched with all the values of right table.

SYNTAX: select cname from firsttablename
cross join secondtable name;



eid	fname	pid	pname
04	vijay	234	shirt
04	vijay	657	top
04	vijay	865	tshirt
04	vijay	378	jeans
7369	sanjay	234	shirt
7369	sanjay	657	top
7369	sanjay	865	tshirt
7369	sanjay	378	jeans

Relation stands for table in DB.

PAGE No.	
DATE	/ /

interesting

Relational DB design

Redundancy of data \rightarrow some piece of information in a table will have duplicate values. Such a data is termed as redundant data.

Anomalies in DB :-
 (i) insertion anomaly \rightarrow if we know some part of a data, and other part of a data is absent, such an information cannot be inserted in a table.

(ii) deletion anomaly \rightarrow if we delete unwanted data, along with deletion of unwanted data, wanted data can also be lost.

(iii) Updation/modification anomaly \rightarrow if we are going to update redundant data, changes must be affected in all the places.

Normalization

Normalization is the process in which redundant data can be removed by eliminating insertion, updation and deletion anomalies.

interesting

Functional dependency

It describes constraints between two attributes. Which says that:-
 for every value of x , we should have exact 1 value of y .

Emp Id	name	Salary
1	Aali	16000
2	Manoj	16000
3	Sandip	10000
4	Vikas	10000
5	Manoj	16000

In this relation X and Y are two attributes

$$X \longrightarrow Y$$

In this case x is functionally determined y if

In this case X is functionally determined Y if
 - for every value of X , we have exactly one value of Y .

$X \rightarrow$ determinant

$Y \rightarrow$ functionally dependent

very interesting

Duplication is allowed

id	name
(x)	(y)

name	sal
(x)	(y)

we have F.D

For salary table to name (Sal \rightarrow name) no F.O

Las 16000 → Adi
→ Manoj

Q.

$$D \rightarrow A$$
$$(D_2 \rightarrow A_1, A_2)$$

$D \rightarrow B$

(F.D)

A_i	B	C	D
A_1	B_1	C_1	D_1
A_1	B_2	C_1	D_2
A_2	B_2	C_2	D_2
A_2	B_2	C_2	D_3
A_3	B_3	C_2	D_4

$$C \rightarrow A \quad \left(\begin{array}{l} C_2 \rightarrow A_2 \\ \quad \rightarrow A_3 \end{array} \right)$$
$$C \rightarrow B \quad \begin{pmatrix} C_1 \rightarrow B_1 \\ \quad \quad \quad \searrow \\ \quad \quad \quad B_2 \\ C_2 \rightarrow B_2 \\ \quad \quad \quad \searrow \\ \quad \quad \quad B_3 \end{pmatrix}$$
$$A \rightarrow B \quad \left(\begin{array}{c} A_1 \rightarrow B_1 \\ \quad \searrow \\ \quad B_2 \end{array} \right)$$
$$B \rightarrow A \quad (A^2)$$
$$B \rightarrow C \quad (B_2 \rightarrow C_1, C_2)$$
$$B \rightarrow 0 \quad \left(\begin{array}{l} B_2 \rightarrow D_2 \\ \quad \rightarrow D_3 \end{array} \right)$$
$$D \rightarrow C$$
$$\begin{pmatrix} 0_2 \rightarrow C_1 \\ \rightarrow C_2 \end{pmatrix}$$

$$A \rightarrow C \quad (A \text{ NOT } F.D)$$

$$A \rightarrow D \quad \left(\begin{array}{l} A_1 \rightarrow D_1 \\ \quad \rightarrow D_2 \\ A_2 \rightarrow D_2 \\ \quad \rightarrow D_2 \end{array} \right)$$

Types of functional dependency

- ① Fully Functional Dependency and Partial Functional Dependency
- ② Transitive F.D and No Transitive F.D
- ③ Single valued F.D and Multivalued F.D

name	address	course	fee	grade
p	Andheri	ADBS	700	A
q	Parel	DBMS	800	B
r	Dadar	MP	500	A
s	Thane	DBMS	800	A
p	Andheri	TCS	900	BA
t	Vihar	TCS	900	B

Primekey attribute \rightarrow attribute part of prime key

Fully functional dependency \rightarrow if we have any non primekey attribute which is used to functionally determine dependent upon all the primekey attributes, then it becomes fully functional dependency.

Partial functional dependency \rightarrow if any non primekey attribute can be identified by any of the primekey attribute, it becomes partial functional dependency.

Transitive F.D \rightarrow if $x \rightarrow y$ and $y \rightarrow z$ then $x \rightarrow z$ is true for transitive F.D

and
No Transitive F.D

if F.D is not transitive, it becomes No transitive F.D

id	sem	hostel
501	V	H-5
400	IV	H-4
513	V	H-5

eg $id \rightarrow sem$

$sem \rightarrow$ hostel

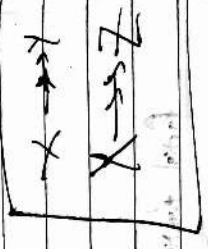
than

$sem \rightarrow$ hostel

(Transitive F.D)

Single valued F.D \rightarrow for every value of X, we have exactly one value of Y.

Multivalued F.D \rightarrow for every value of X, we have multiple values of Y.



Representation: $X \rightarrow Y$

multivalued dependency

Decide whether the following are FD or MVD

id	teacher	class	days
1	Sam	comp	3
12	John	comp	6
11	Sam	Phy	3
12	John	mech	6
13	Mike	mech	2

$id \rightarrow teacher$

(single valued)

$id \rightarrow class$

$id \rightarrow$ not there

(hence not multivalued)

$class \rightarrow teacher$

$days \rightarrow$ (multivalued)

interesting class \rightarrow teacher
John class \rightarrow days
no attach to attribute

two attributes

Normalization

- 1NF
- 2NF
- 3NF
- BCNF
- 4NF

① First Normal Form (1NF)

All the values present in a table must be atomic (single)

② Second Normal Form (2NF)

- a) Table must be present in 1NF
- b) No partial λ dependency is present.

③ Third Normal Form (3NF)

- a) Table must be present in 2NF
- b) Transitive dependency must be removed.

④ BCNF (Boyce - Codd NF)

- a) Table must be in 3NF
- b) All the determinants must be primary key.

⑤ Fourth Normal Form (4NF)

- a) Table must be in BCNF
- b) Multivalued dependency must be removed

g:-

name	address	course	grade	fee	sem	hostel	duration
xyz	dadar	C1, sec 2	a ₁ , b ₁	7000	V1	H7	10
xyz	dadar	C2	b ₁	8000	V1	H2	11
pqr	panel	C2	b ₁	5000	V1	H4	06
abc	gadhori	C2, sec 1	b ₁	8000	V1	H5	09
abc	gadhori	C1	b ₁	9000	V1	H3	09

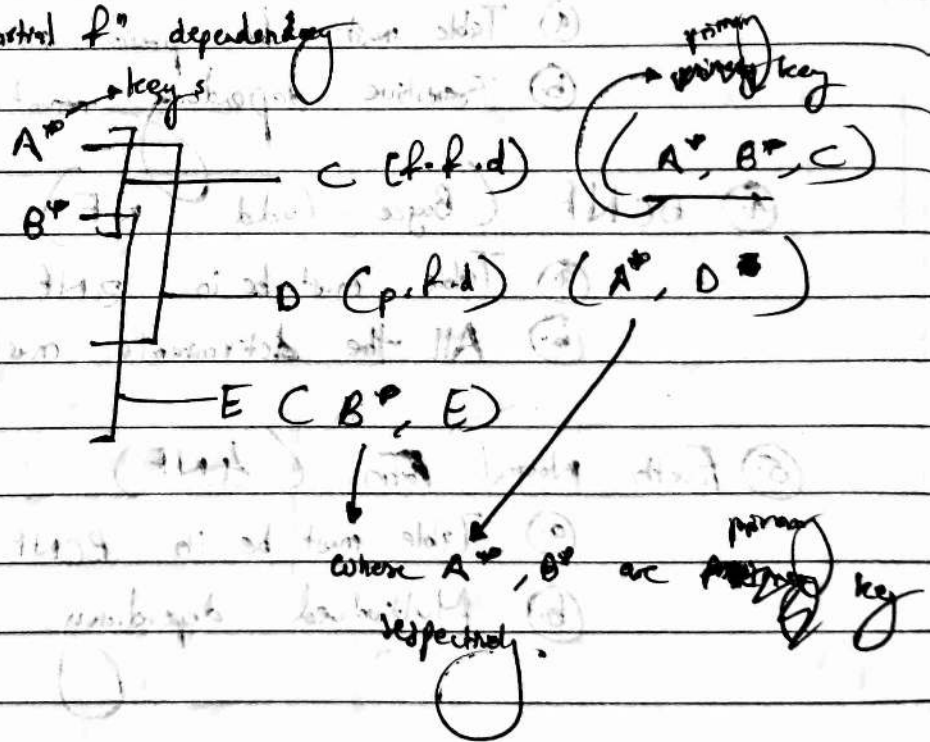
values added for INF as all the values missing vals are put in INF

INF → we decided candidate key as $\{ \text{name}, \text{course} \}$
 $A^* \quad B^*$

2NF → It must be in 1NF

No partial f^o dependencies

Rules:-



name → address (p.f.d)
 course → fee, sem (p.f.d)

$\{ \text{name}, \text{course} \} \rightarrow \text{grade, hostel, duration (h.f.d)}$

①

name	address	
xyz	dadar	
PQR	park	
abc	andheri	

ck = { name }

②

course	fee	sem
C1	9000	V1
C3	8000	III
C2	5000	V
C2	5000	V
C1	9000	V1

ck = { course }

③

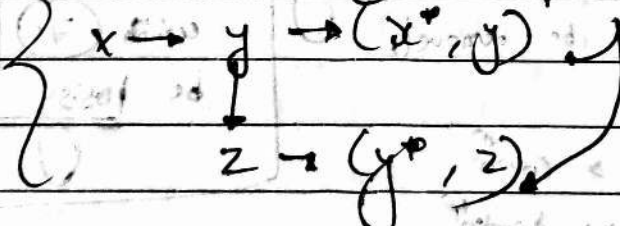
name	course	grade	hostel	duration
xyz	C1	97	H7	10
xyz	C3	61	H3	11
PQR	C2	61	H5	06
ABC	C2	61	H5	09
ABC	C1	61	H3	09

ck = { name, course }

3NF → Table must be in 2NF

No transitive dependency should be present

rule: → separate table for



(1st and 3rd already in 3NF)

pk (course) → fee

pk (fee) → sem

④

course	fee
C1	9000
C3	8000
C2	5000

⑤

fee	sem
9000	V1
8000	III
5000	V

After performing 3NF, we have ①, ③, ④ and ⑤

BCNF \rightarrow (1), (4) and (5) table are in BCNF.

(3) table is not in BCNF as
 $\text{hostel} \rightarrow \text{grade}$
 is a functional dependency.

AND

hostel is not a candidate key.

\therefore (3) table is divided into 2 parts.

(6)				(7)	
name	course	duration	hostel	hostel	grade
XYZ	C1	10	H7	H7	A7
XYZ	C3	11	H3	H3	B1
PQR	C2	06	H4	H4	B1
ABC	C2	09	H5	H5	B1
ABC	C1	09	H3	H3	B1

check?
 duration \rightarrow name
 Assumed not in BCNF to show 4NF

ck = {name, course}

BCNF = {1, 4, 5, 6, 7}

4NF \rightarrow ~~transitive~~ multivalued dependency must be removed;
 as need hostel box to relate with (7) else it will be lossy normalization

name \twoheadrightarrow course
 name \twoheadrightarrow duration
 name \twoheadrightarrow hostel

(8)

name	course
XYZ	C1
XYZ	C3
PQR	C2
ABC	C2
ABC	C1

(9)

name	duration
XYZ	10
XYZ	11
PQR	06
ABC	09

(10)

name	hostel
XYZ	H7
XYZ	H3
PQR	H4
ABC	H5
ABC	H3

4NF = {1, 4, 5, 7, 8, 9, 10}

Transaction management and concurrency

The amount of work that is performed in a DB is ~~measured~~ measured by a unit called transaction.

Three operations are performed on transaction. These are:-

- $Read(x)$ → transfers value of x from secondary to main memory and then reads the value of x .
- $Modify(x)$ → changes or modifies the values of x present in MM.
- $Write(x)$ → the modified value is written back from MM to secondary storage.

Example: T_1 transaction is run parallel to T_2 transaction.

T_1 : $A = 500$; $read(A)$; $A = A - 50$; $write(A)$; $A = 450$

T_2 : $B = 1000$; $read(B)$; $B = B + 50$; $write(B)$; $B = 1050$

write(B); // 1050

Transaction Properties

(ACID)

Atomicity

Consistency

Isolation

Durability

interesting

(check if ~~structure~~ implemented using interest?)

Job of
Transaction manager
DB
programmer

Atomicity → all the operations must be performed completely or none of them are performed.

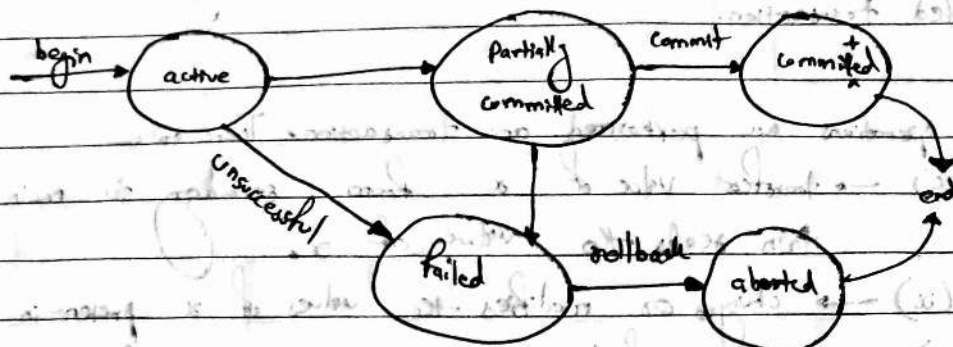
Consistency → the state before and after the execution of transaction should be same.

Concurrency component → $\sum(A+B)$ must be preserved

Isolation → All the transactions are running in isolated manner without affecting each other.

Recovery manager → Durability → Even if power failure occurs, DB is present in the system.

Transaction states



begin:- indicates start of transaction

partially committed:- if transaction reached to the second last command of the transaction. (before commit)

failed:- transaction can fail in between or it can fail in partially committed state. Such ^{transactions} enter in failed state.

committed:- transaction successfully finished execution.

aborted:- failed transactions are in aborted state. Two operations are performed:-

1. Restart:- hardware/software failure

a. Kill:- data is missing

end:- know the execution of transaction.

schedules

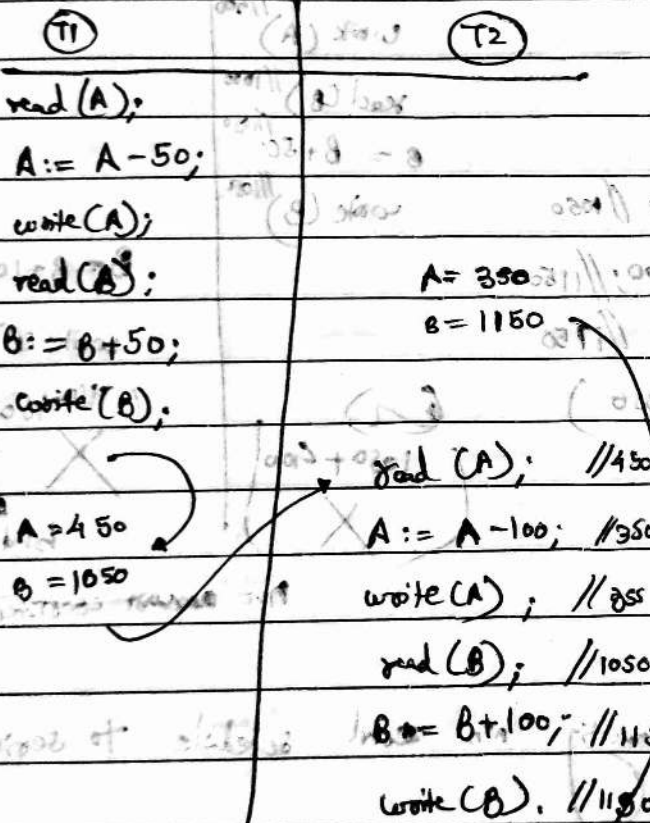
- When two or more transactions are combined together, schedules are created.
- Schedules describe the order of execution of transactions
- Two types: -
 - i. Serial
 - ii. Concurrent

→ Always consistent

- Serial schedule → All the transactions are executed in serial manner. (one after another)

interesting

IF serial schedule is there in the DB, consistency is always provided.



Two conditions:-

- ① dependency
- ② without dependency

$$\Sigma(A+B) = 1800 \rightarrow \text{is preserved}$$

both are isolated

EXERCISE

also called Non Serial Schedule

Consistency may or may not be possible

Concurrent Schedules:-

Transactions are running concurrently with context-switching.

T_1	T_2	T_1	T_2
read(A); //500		read(A); //500	
$A := A - 50$; //450		$A := A - 50$; //450	
write(A); //450			read(A) //500
	read(A); //450		$A := A - 100$; //400
	$A := A - 100$; //350		write(A) //400
	write(A); //350		read(B) //1000
read(B); //1000		write(A) //400	
$B := B + 50$; //1050		read(B) //1000	
write(B); //1050		$B := B + 50$; //1050	
	read(B); //1050		write(B) //1050
	$B := B + 100$; //1150		
	write(B); //1150		

is this serial execution?

(1050 + 450)

(1150 + 350)

(1050 + 400)

(1150 + 400)

(1150 + 450)

Serizability \rightarrow process of converting non serial schedule to serial schedule.

- (i) Conflict
- (ii) View

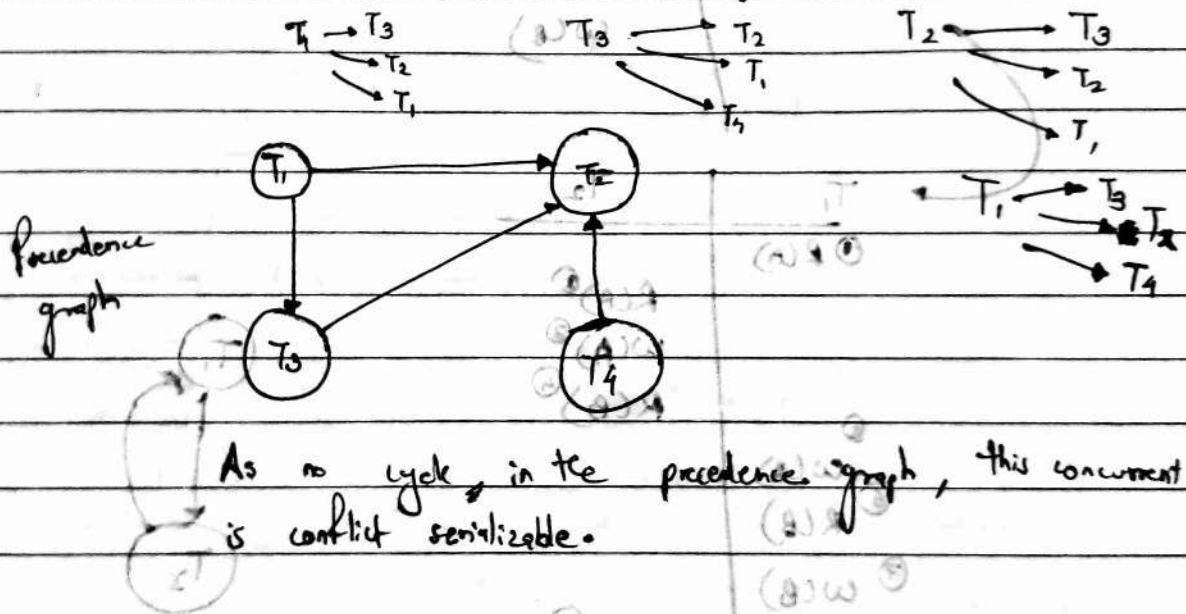
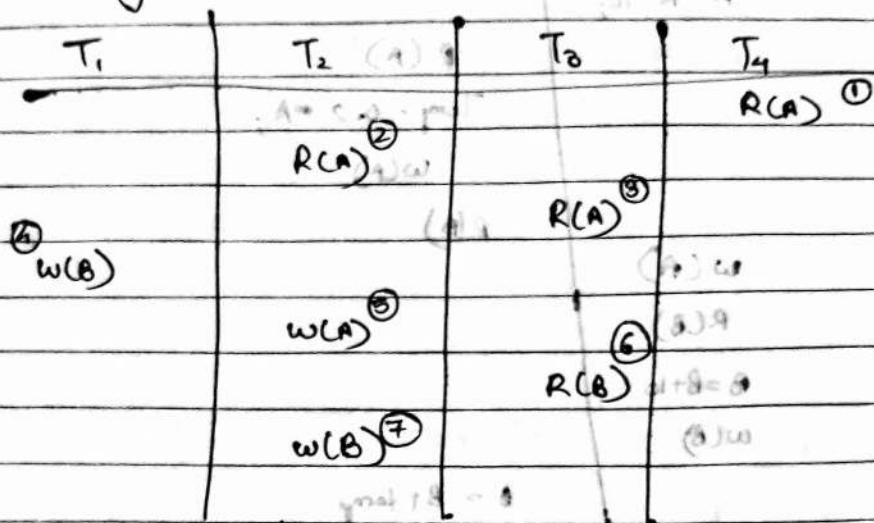
Conflict \rightarrow obtained by swapping non conflicting operations.

Rules for conflicting operations:-

- both the operations must belong to different transactions.
- both the operations must be on same data item.

③ Atleast one of the two operations is write operation.

Q. Check if the given schedule 's' is conflict serializable or not



IT \leftarrow ST
 (A)W (A)R
 (A)W (A)R

Q.

T_1 T_2

$R(A)$;
 $A = A - 10;$

$R(A)$
 $Temp = 0.2 * A;$
 $W(A)$

$W(A)$

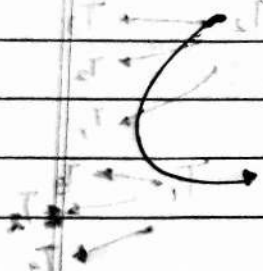
$R(B)$

$B = B + 10$

$W(B)$

$B = B + Temp$

$W(B)$



T_1

① $R(A)$

$R(A)$ ②

$W(A)$ ③

$R(B)$ ④

⑤ $W(A)$

⑥ $R(B)$

⑦ $W(B)$

$W(B)$ ⑧

$T_1 \rightarrow T_2$

$R(A) W(A)$

$T_2 \rightarrow T_1$

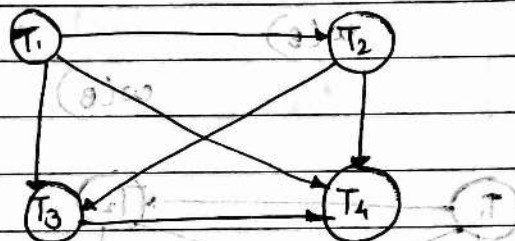
$R(A) W(A)$

$R(B) W(B)$



As cycle, we do not have conflict serializability.

S			
T ₁	T ₂	T ₃	T ₄
RCA)			
	RCA)		
		RCA)	
			RCA)
wCB)			
	wCB)		
		wCB)	
			wCB)



As no cycle, it is conflict serializable.

View serializability → Any non serial serial schedule is found to be view serializable if it satisfies following 3 rules:-

① If schedule is conflict serializable, then it is 100% view serializable.

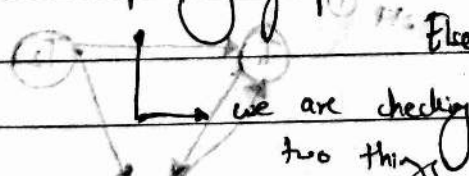
Else it may or may not be view serializable.

② No blind-write → no view serializability.

If yes → may or may not be view serializability.

③ Draw dependency graph → If cycle → no view serializability.

Else if no cycle → view serializability.



we are checking two things
(not write, not write)

no write, no write

T is having 2 children

3 children

Block write \rightarrow Transition is not ready and starts writing. Block

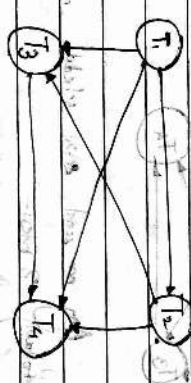
Q. Consider the given schedule S and check whether it is view serializable or not.

T_1	T_2	T_4
R(A)	R(A)	R(A)
R(A)	R(A)	R(A)

$w(B)$ $w(B)$ $w(B)$

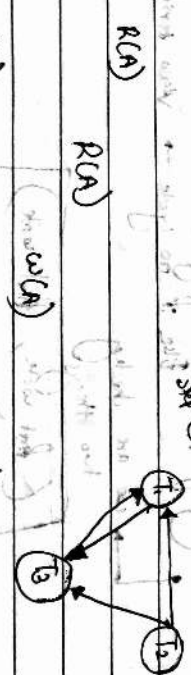
if it is serial on SA

Step 1:-



As in precedence graph cycles are not present, it is conflict serializable. Therefore it is also view serializable.

Q. T_1, T_2, T_3

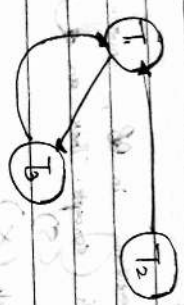


As cycle is present, it may or may not be view serializable.

Step 2: As block write is present in T_3 ,

it may or may not be view serializable.

step ③



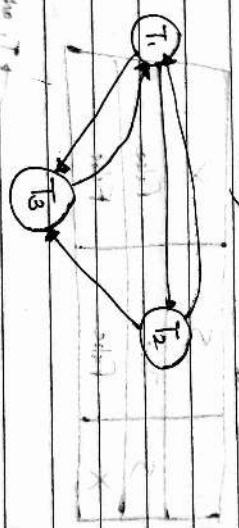
As cycle is present in dependency graph, it is not view serializable.

a. $T_1 \quad T_2 \quad T_3$

RCA)

COCA)

WCA)



step ① Cycle is present or may not be present - serializable.
step ② $T_2 \rightarrow$ biconic is present,
may or may not be view serializable for A

step ③



no cycle,
view serializable

(A) lock

(B) signal

(C) wait

(A) lock

(B) signal

(C) wait

Concurrency control in DBMS

In order to maintain consistency in DB, concurrency control is used.

It provides locking concepts. There are of three types:-

- shared lock (S)
- exclusive lock (X)

used for read operation. One shared lock is acquired, no 'exclusive lock (X)' can be obtained. (But shared lock can be obtained).

used for reading and writing operation. If exclusive lock is obtained, no other lock can be acquired (No shared lock, No exclusive lock).

Can we obtain (A) (C)

If we have

S	S	X
S	True	False
X	False	False

Copy the schedule and in two which two operations are given. We have 2 queries:-

A → 1000
B → 500

order lock
A-lock (A)
read (A)
write (A)
unlock (A)

Serially
A = A + 500
write (A)
unlock (A)
S-lock (B)
read (B)
unlock (B)
display (A+B)

B = B + 500
unlock (B)

Cascaded rollback

work



DATE	/	/
------	---	---

LOCKS ARE NOT SPECIFIC TO A PARTICULAR TRANSACTION

T_1 T_2

X-lock(A)

read(A)

$A = A - 500$

write(A)

unlock(A)

S-lock(A)

read(A)

$A = A - 500$

unlock(A)

S-lock(B)

read(B)

unlock(B)

display(A+B)

to prevent
down to it should

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10 (A) lock-X
- 11 (A) lock
- 12 (A) lock
- 13 (A) lock
- 14 (A) X-lock(B)
- 15 (A) read(B)
- 16 (A) $A = B - 500$
- 17 (A) write(B)
- 18 (A) unlock(B)
- 19 unlock(A)

disadvantages of unlock(A) at line 5

- Consistency is not maintained

interesting cascaded rollback: if unlock(A) is at line 5 and there is an error in transaction T_1 , then T_1 has to be

rolled back. As S-lock(A) gets acquired as

unlock(A), it creates a cascading effect. and T_2 has to be rolled back too.

By putting unlock(A) at line 19, we prevent both inconsistent state and cascaded rollback.

caused rollback because of transaction T_1 failure, T_1 is rolled back, but T_2 is reading the value which is given by transaction T_1 and therefore T_2 needs to be rolled back.

Case

T_1
 $X\text{-lock}(A)$
 $read(A)$
 $A = A - 500$
 $write(A)$

$X\text{-lock}(B)$
 $read(B)$
 $A + 500$
 $write(B)$
 $unlock(B)$

$unlock(A)$

$S\text{-lock}(B)$

$read(B)$

$S\text{-lock}(A)$

$read(A)$

$unlock(A)$

$unlock(B)$

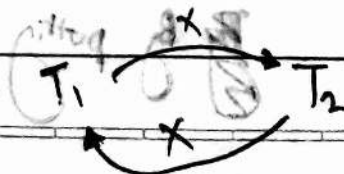
$display(A+B)$

In the above case, circular deadlock is seen.

$S\text{-lock}(B)$ is read locked mode, which prevents $X\text{-lock}(B)$

$S\text{-lock}(A)$ cannot be locked as $X\text{-lock}(A)$ is yet to

be unlocked in T_1 .
 we have deadlock.



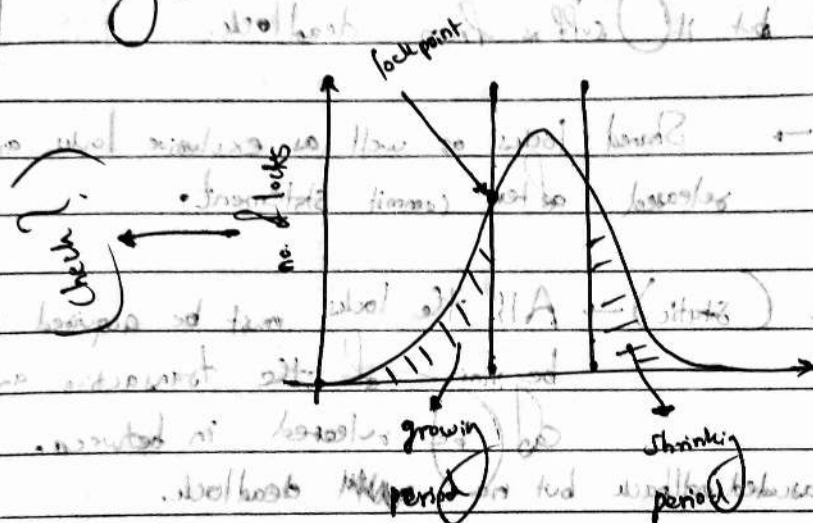
2PL (2 phase locking) Protocol

Two phases are there:-

- growing phase
- shrinking phase

Growing phase \rightarrow all the locks are acquired and none of them is released.

Shrinking phase \rightarrow all the locks are released and none of them are acquired.



lockpoint \rightarrow point at which last lock is acquired.

Three types of 2PL:-

- Strict 2PL \rightarrow all exclusive locks are released only when transaction permits. (after commit)
- Shared lock can be released in between.

(A) T_3

(B) T_4

(A) S-lock (A)

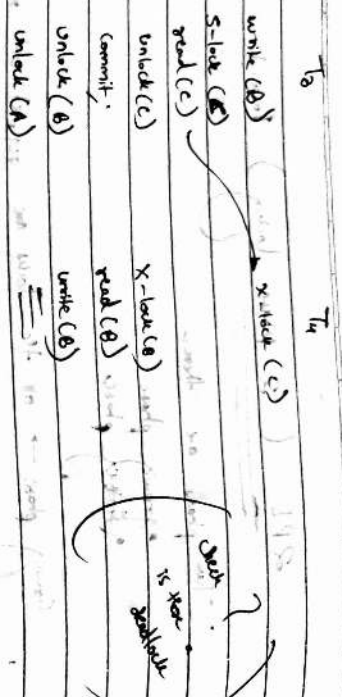
(B) shared

read (A)

(B) exclusive

X-lock (B)

read (A)



Strict 2 phase locking protocol does not suffer from cascaded rollback but it suffers from deadlock.

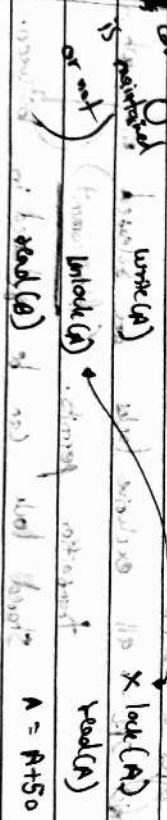
Does not suffer from cascaded rollback
 (a) Rigorous 2PL → Shared locks as well as exclusive locks are released after commit statement.

(b) Conservative 2PL (static) → All the locks must be acquired at the beginning of the transaction and released in between.

We have cascaded/rollback but no right deadlock.



(check)
 A = A + 50



write(A)

unlock(A)

unlock(B)

(A) lock

Timestamp ordering protocol

It is for running each transaction timestamp value is given. Depends upon timestamp, the execution of the transaction takes place.

Two rules:-

① Transaction T_i issues read (R)

② If $TS(T_i) < \text{write-timestamp}(Q)$ (interest)

read operation is rejected and T_i rollback

③ If $TS(T_i) \geq \text{write-timestamp}(Q)$

read operation is performed and $\text{read-timestamp}(Q)$

$\text{read-timestamp} = TS(T_i)$

④ If T_i issues write (W)

⑤ If $TS(T_i) < \text{read-timestamp}(Q)$ OR $TS(T_i) < \text{write-timestamp}(Q)$

$TS(T_i) < \text{write-timestamp}(Q)$ or $TS(T_i) < \text{read-timestamp}(Q)$

rejected, rollback $TS(T_i) = (Q) \cup TS(T_i)$

⑥ otherwise accepted

$\text{write-timestamp}(Q) = TS(T_i)$

rollback

$$RTS(A) = 10$$

$$WTS(A) = 10.05$$

$$RTS(B) = 10.05$$

$$WTS(B) = 10.05$$

T_1	T_2	description
$T_s = 10$	$T_s = 10.05$	
read (B)	read (B)	$TS(T_1) \geq WTS(B)$ $10 \geq 0$ $time(B) = 10$
	$b = b - 20$	
	write (B)	$TS(T_2) \geq WTS(B)$ $10.05 \geq 10$ $time(B) = 10.05$
read (A)	read (A)	
print (A+B)	$A = A + 30$	
	write (A)	$WTS(B) = TS(T_2)$ $= 10.05$
	print (A+B)	
$TS(T_1) \geq WTS(A)$ $10 \geq 0$ $RTS(A) = 10$		
$TS(T_1) \geq WTS(A)$ $10.05 \geq 10$ $RTS(A) = 10.05$	$WTS(A) = TS(T_2)$ $= 10.05$	

As all operations are performed concurrently under the given timestamp value, this schedule ensures freedom from deadlock.

$$RTS(A) = 10.05$$

$$WTS(A) = 10.05$$

$$RTS(B) = 10.05$$

$$WTS(B) = 10.05$$

PAGE NO.	
DATE	/ /

T_1
 $T_s = 10$

T_2
 $T_s = 10.05$

$TS(T_1) = 10$ ← read(B)

$RTS(B) = 0$
 $WTS(B) = 0$

$RTS(A) = 10$

read(A)

$b = b - 20;$

write(B);

read(A);

$a = a + 20;$

write(A);

print(A+B);

$TS(T_2) = 10.05$

$WTS(B) = 0$

$RTS(B) = 10.05$

$TS(T_2) = 10.05$

$WTS(A) = 0$

$RTS(A) = 10.05$

Failed Read
~~write(B)~~

$WTS(B) = 10.05$

read(A)

print(A+B);

$TS(T_1) = 10$

$WTS(A) = 10.05$

$RTS(A) = 10.05$

$TS(T_2) = 10.05$

$WTS(A) = 0$

$RTS(A) = 0$

$RTS(A) = 10.05$

As $WTS(A) > TS(T_1)$

Cannot read

Rejected

Rollback

As T_1 is rolled back,

T_1 is rejected and it has to

start with new timestamp value.

$TS(T_2) = 10.05$

$WTS(A) = 0$

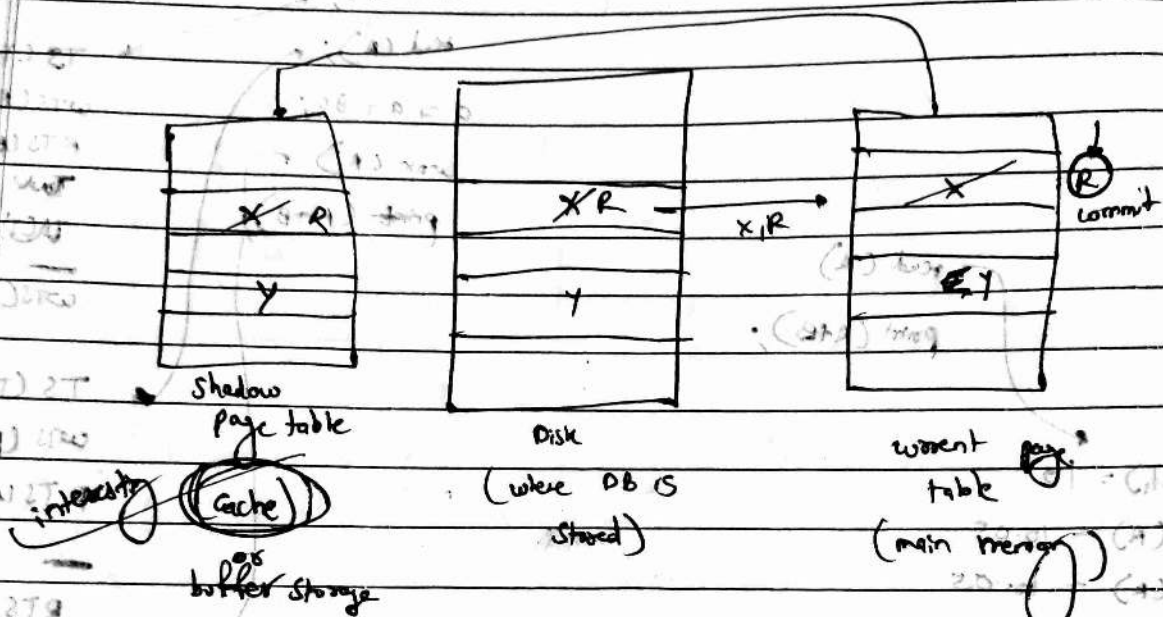
$RTS(A) = 10.05$

$WTS(A) = 10.05$

Shadow paging

This is the alternative for log based recovery where two page tables are maintained: —

- shadow page table
- current page table



In case contents of current and shadow page table are same. If we want to perform operation on data item X, it gets transferred from disk to current page table, and current page table copies it to shadow page table.

If transaction is not committed, data can be recovered from shadow page table and if transaction is committed, Current page table copies data item to Shadow PT and Shadow PT transfers it to disk.

Deadlock is a situation where one transaction is waiting for another and vice versa. Therefore no transaction can proceed further with its normal execution.

10/11/2017

(wfa)

If cycle is present, transaction is in deadlock state,
Else if no cycle, transaction is deadlock free.

in deadlock state

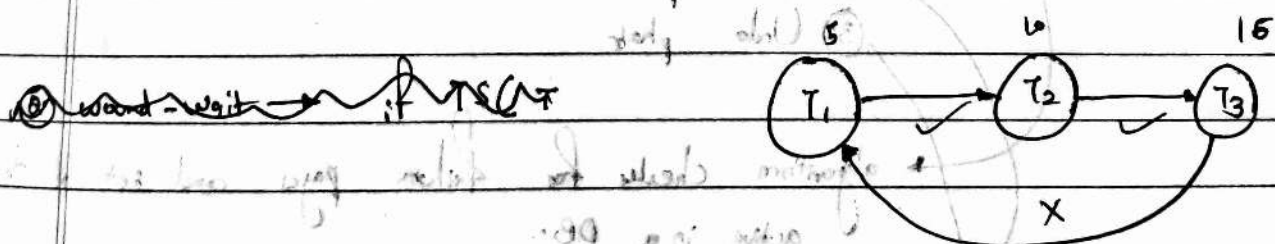
in deadlock state

$$\frac{(T)T \leq (P)PT}{(Hence \ P)}$$

→ T_i waits

else

Ti rollback


$$T_S(T_1) < T_S(T_2) \quad (T_1 \text{ wants})$$
$$TS(T_2) < TS(T_3) \quad (T_2 \text{ waits})$$
$$TS(\tau_0) \neq TS(\tau_1)$$

(T3 rolled back)

operation rejected with
interlocking some system
delay.

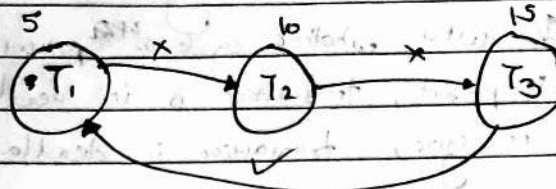
With some
~~random~~
fine as

operation is accepted.

② would wait \rightarrow if $TS(T_i) > TS(T_j)$

T_i waits
else

T_j rollback



$TS(T_2) \neq TS(T_3)$

$TS(T_2)$ rolled back

$TS(T_2) \neq TS(T_3)$

$(T_3$ rolled back)

$TS(T_3) > TS(T_1)$

$(T_3$ waits)

ARIES \rightarrow algorithm for recovery and isolation exploiting semantics

① Analysis phase

② Redo phase

③ Undo phase

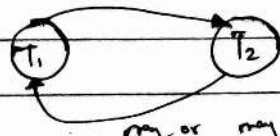
\rightarrow algorithm checks for failure pages and set of transactions active in a DB.

\rightarrow it reapplies updates from the log to the DB
 \rightarrow it is done only for committed transactions.

\rightarrow the log is scanned in backward direction and the no. of transactions which are active at time of transaction crash are restored with old values.

Q.

T_1	T_2
$R(A)$	
$W(A)$	$R(A)$
$R(B)$	$W(A)$
$W(B)$	$R(B)$
	$W(B)$



may or may not be view serializable
no blind write,
not view serializable

Q.

